



ISSN 0933-9434 | November/Dezember 1988 | 2.Jg./Heft 6
ÖS 150,- | DM 19,80 | sfr 19,80

Tips zum Datenaustausch in Windows

Stapelverarbeitung unter Windows

Bildschirminhalte auf Knopfdruck speichern

Bildschirmausdruck in Assembler

TSR

Speicherresidente Programme in Assembler und C



Hörst du, der druckt alles, was auf den Tisch kommt! Wie, du hörst nichts...?

Man kann den RX 7100 zwar leicht überhören, aber seine Qualitäten sind nicht zu übersehen: Sie können es sich jetzt leicht machen und eine zweite Papierkassette einsetzen. Dann ist der schnelle Wechsel kein Problem. Vom Geschäftspapier zum Formular, vom Erst- zum Zweitbogen. Das wird dann mit 5 Seiten pro Minute gestochen scharf bedruckt. Sogar Postkartenkarton bis zu 185 g/m²! Der RX 7100 ist ein komfortabler Tischdrucker mit minimalen Ausmaßen und maximaler Leistung. Seine LED-Technologie sichert Ihnen hervorragende Qualität und Zuverlässigkeit. Dieser Drucker läßt sich aus nahezu jeder modernen Software ansprechen. Das macht ihn zum idealen Partner für Ihre Arbeit am Einzelplatzcomputer oder Schreibsystem, am PC, im Desktop Publishing und für CAD/CAM.

Problemlose Aufrüstung bis zu drei IC-Karten für Schriften- und Drucker-Emulation können gleichzeitig von den Steckkartenschlitzen aufgenommen werden.

Er druckt praktisch alles, was „auf den Tisch kommt".

FUJITSU RX 7100.
Der komfortable Tischdrucker.

Bitte senden Sie mir genauere Informationen über
den FUJITSU RX 7100
das gesamte FUJITSU-Drucker-Programm
Name
Straße
PLZ/Ort
Firma

SC

Coupon an: FUJITSU DEUTSCHLAND GMBH
Rosenheimer Str. 145 · 8000 München 80
Drucker-Telefon-Hotline 089/41301152

FUJITSU

Drucker von Japans Computerhersteller Nr. 1

tm 5370

Ausgabe November/Dezember 1988




TSR

Microsoft Windows

Multiplan 4.0

Microsoft C

Rubriken

TSR - Terminate and Stay Resident:

Speicherresidente Programme in MS-C

Speicherresidente Programme haben einen geheimnisvollen Nimbus. Dies liegt zum Teil daran, daß sie auf Knopfdruck wie von Zauberhand erscheinen und wieder verschwinden, wenn sie nicht mehr benötigt werden. Der TSR-Nimbus beruht jedoch am stärksten auf der Macho-Philosophie, die dahinter steckt. TSR-Programmierer haben eine Haltung, die sich am besten mit dem Satz umschreiben läßt: »Was geht mich das Betriebssystem an, ich programmiere das so, wie ich das will!«. Und sie gehen mit dieser Haltung bis zum Äußersten.

TSR steht für »Terminate and Stay Resident«. Es handelt sich dabei um Programme, die die Fähigkeiten von DOS erweitern. Einige TSR-Programme arbeiten still und heimlich; sie haben eine nützliche Funktion, brauchen jedoch nicht direkt mit dem Benutzer zu kommunizieren. Andere TSR-Programme sind interaktiver; wenn sie merken, daß ihre Aufruftaste (hot key) gedrückt wurde, erscheinen sie auf dem Bildschirm, zeigen Bildschirmdialoge an, führen ihre Dienste aus und verschwinden wieder, wobei der vorherige Zustand des Systems wiederhergestellt wird. Das wohl bekannteste Beispiel für ein TSR-Programm ist SideKick von Borland.

Das erste kommerziell erfolgreiche TSR-Programm war ProKey von RoseSoft, das 1982 vorgestellt wurde, doch es war SideKick, mit dem die TSR-Manie eigentlich begann. Noch heute ist es eines der wichtigsten TSR-Programme für DOS. Die Entwickler dieses Programms erkannten lange vor allen anderen, daß DOS auf Wachstum ausgerichtet ist.

Durch Aufruf von Interrupt 27h oder der DOS-Funktion 31h kann sich ein Programm beenden und zu DOS zurückkehren, jedoch im Speicher verbleiben. (Normalerweise verwendet DOS den Speicherbereich eines Programms nach seiner Beendigung wieder für andere Programme.) Ein solches Programm ist dann eine Erweiterung von DOS, denn wie DOS existiert es, um Hard- oder Softwareinterrupte zu behandeln.

Der einzige Unterschied zwischen einem TSR-Programm und DOS ist, das es sich bei den DOS-Funktionen meistens um langweilige Dinge wie die Datei- oder Tastaturein-/ausgabe handelt. Diese Funktionen sind für die Arbeit mit einem PC lebenswichtig, sie sind jedoch einfach nicht eindruckvoll genug, um seine Benutzer zu Freudensprüngen zu veranlassen. Pop-Up-Programme wie SideKick sind die Dinge, aus denen Träume gemacht sind.

TSR-Programme in C

TSR-Programme sind das letzte As im Ärmel derjenigen, die Assembler immer noch für die beste Sprache halten. Mit UNIX ist gezeigt worden, daß sogar ein effektives Betriebssystem überwiegend in einer höheren Programmiersprache geschrieben werden kann. Und auch wenn die erste Generation von PC-Programmen hauptsächlich in Assembler geschrieben worden ist, werden heute die meisten Programme in C entwickelt und nur noch die absolut notwendigen Teile in Assembler. Doch im Reich der TSR-Programme ist Assembler noch der König und es gibt gute Gründe dafür, daß Assembler seine Herrschaft fortsetzen wird.

Dieser Artikel soll Ihnen demonstrieren, wie TSR-Programme in C geschrieben werden; dabei wird darauf eingegangen, was sie dabei gewinnen, aber auch, was Sie verlieren. Mit C müssen Sie genauso viel wissen; auch wenn Sie TSR-Programme in C programmieren, müssen sie wissen, wie die PC-Hardware, DOS, Interrupte und Stacks funktionieren. Darüber hinaus wurden die meisten veröffentlichten Informationen über TSR-Programme von Assemblerprogrammierern geschrieben. Natürlich brauchen sie keinen Assembler zu kaufen und auch nicht zu lernen, wie er funktioniert, doch ganz gleich, ob sie ein TSR-Programm in C oder Assembler programmieren, es ist nicht so einfach, wie ein »Hallo Welt«-Programm.

Wie Sie sich sicher denken können, benötigen TSR-Programme in C mehr Speicherplatz als entsprechende Assemblerprogramme. Dafür gibt es mehrere Gründe. Einer ist, daß selbst gute Compiler wie der von Microsoft nicht so kompakten Code erzeugen, wie ein erstklassiger Programmierer. Für die meisten Programme ist der Unterschied unwichtig, doch bei einem TSR-Programm kommt es auf jedes Byte an und jedes Byte zuviel wird von Zeitschriftenredakteuren und Benutzern kritisch beäugt. Der Hauptgrund für die Größe von C-TSR-Programmen ist, daß die meisten C-Programme eine Menge Routinen aus der C-Library verwenden und in das ausführbare Programm aufnehmen, meist mehr, als man glaubt. Sehen Sie sich einmal die MAP-Datei von etwas ganz einfachem, zum Beispiel dem Programm »Hallo Welt«, an, und Sie sehen sofort, was ich meine.

Sie können sich auch denken, daß C-TSR-Programme dazu tendieren, langsamer zu sein, als solche in Assembler. Das ist meist kein Problem, denn die meisten bevorzugen für fast alle Anwendungen Hochsprachen.

Der Vorteil beim Einsatz von C für TSR-Programme gleicht den Vorteilen von C in anderen Bereichen. C ist einfacher zu schreiben, zu testen und zu warten als Assembler. Auch gibt es mehr gute C-Programmierer als gute Assemblerprogrammierer. Und viele ausgezeichnete Programmierer, die wohl auch in Assembler programmieren können, ziehen die Produktivitätssteigerung bei der Verwendung von C vor. Wenn Sie das nächste SideKick schreiben wollen, ist die traditionelle Vorgehensweise und Programmierung in Assembler wahrscheinlich am besten; doch für weniger ehrgeizige Projekte, zum Beispiel ein TSR-Programm für den hausinternen Einsatz, sollte C ernsthaft in Erwägung gezogen werden.

```
void interrupt far intfarfn(es, ds, di, si, bp, sp, bx, dx,
                            cx, ax, ip, cs, flags)
unsigned es, ds, di, si, bp, sp, bx, dx, cx, ax, ip, cs, flags;
{
}
```

```
        PUBLIC  intfarfn
_intfarfn       PROC FAR
        push    ax
        push    cx
        push    dx
        push    bx
        push    sp
        push    bp
        push    si
        push    di
        push    ds
        push    es
        mov     ax,DGROUP
        mov     ds,ax
        ASSUME DS: DGROUP
        cld

;       es = 0
;       ds = 2
;       di = 4
;       si = 6
;       bp = 8
;       sp = 10
;       bx = 12
;       dx = 14
;       cx = 16
;       ax = 18
;       ip = 20
;       cs = 22
;       flags = 24

        pop     es
        pop     ds
        ASSUME DS: DGROUP
        pop     di
        pop     si
        pop     bp
        pop     bx
        pop     bx
        pop     dx
        pop     cx
        pop     ax
        iret
_intfarfn       ENDP
```

Listing 1: *Ein Beispiel für eine C-Interruptroutine und der dafür erzeugte Assemblercode.*

Ich werde im Detail auf die Möglichkeiten und Techniken von Microsoft C für die Programmierung von TSR-Programmen eingehen und einige oft mißverstandene C-Möglichkeiten wie zum Beispiel Zeiger auf Funktionen erläutern. Mein Ziel ist es nicht, einen allgemeingültigen Leitfaden für die Techniken der Programmierung von TSR-Programmen zu geben, doch viele Informationen dieser Art sind natürlich vorhanden. Es wird ein Beispielprogramm entwickelt, das auf Knopfdruck aktiv wird und die Zeit anzeigt. Es ist wegen seiner Einfachheit und Kürze ein gutes Beispiel, jedoch kein umfassendes TSR-Programm, das alle Möglichkeiten demonstriert.

Obwohl Microsoft C über alle Fähigkeiten zur Erstellung eines TSR-Programms verfügt, ist es kein einfacher Einstieg in die TSR-Welt. Wenn man Microsoft C verwendet, muß man seine eigenen Sicherheitsstrategien für die Probleme von DOS und die Koexistenz mit anderen TSR-Programmen entwickeln. Eine Alternative dazu besteht darin, sich eine Funktionsbibliothek wie zum Beispiel /*resident C*/ zuzulegen. /*resident C*/ enthält etwa ein Dutzend Unterroutinen, die einem bei der Erstellung von TSR-Programmen behilflich sind. (/*resident C*/ wird angeboten von Essential Software, Maplewood, New Jersey, Tel.: 001 (201) 762-6965; in Deutschland ist das Paket über mehrere Importeure erhältlich.) Obwohl Essential Software die meisten Details behandelt, können die Informationen in diesem Artikel für Sie auch nützlich sein, wenn Sie sich /*resident C*/ anschauen.

Traditionelle TSR-Programme

DOS ist dafür bekannt, daß viele Anwendungsprogramme sich nicht an die Regeln halten. In leistungsfähigeren und besser kontrollierten Betriebssystemen wie UNIX und MS OS/2 werden die Programme dazu gezwungen, sich innerhalb des Modells zu bewegen, das vom System vorgegeben wird. Doch fast alle PC-Programme gehen die Sache etwas locker an und ignorieren DOS, außer wenn sie auf Dateien zugreifen.

Die Struktur einer normalen DOS-Anwendung sieht so aus, daß sie von DOS in den Speicher geladen wird und DOS dann die Steuerung daran übergibt. Sobald eine Anwendung gestartet ist, geht sie davon aus, daß sie volle Kontrolle über das System hat. Sie wartet ohne etwas zu tun, wenn es nichts zu tun gibt und läuft so lange, bis sie selbst die Steuerung an DOS zurückgibt. Sie übergibt die Kontrolle dazwischen nur an DOS, wenn sie etwas erledigt haben möchte, zum Beispiel den Teil einer Datei lesen oder schreiben.

Von dieser Struktur und diese Annahmen kann bei einem TSR-Programm nicht ausgegangen werden. Zunächst verhalten sich ein normales Programm und ein TSR-Programm gleich. Wie eine normale Anwendung wird ein TSR-Programm von DOS geladen und DOS übergibt die Steuerung an das Programm. Das TSR-Programm initialisiert sich dann und übergibt die Steuerung wieder an DOS, jedoch so, daß es im Speicher stehen bleibt. An dieser Stelle wird es geheimnisvoll. Das TSR-Programm verhält sich nun wie ein Zuschauer; es betrachtet, was alles so passiert, macht selbst jedoch nichts. Wenn seine Dienste jedoch benötigt werden, aktiviert es sich selbst, führt seine Aufgaben aus und begibt sich dann bis zum nächsten Einsatz wieder hinter die Kulissen.

Interrupte

Selbst PC-Neulinge lernen schnell, daß es zwei Arten von Interrupten gibt, Hard- und Software-Interrupte. Hardware-

```
#include <dos.h>
#define DOS_INT 0x21

void far interrupt newint21();      /* DOS */
void (interrupt far * oldint21)();
unsigned long syscalls = 0;

void interrupt far newint21(es, ds, di, si, bp, sp, bx, dx, cx,
                        ax, ip, cs, flags)
unsigned es, ds, di, si, bp, sp, bx, dx, cx, ax, ip, cs, flags;
{
    syscalls++;
    _chain_intr(oldint21);
}

main()
{
    /* alten Int21-Vektor sichern */

    oldint21 = _dos_getvect(DOS_INT);
    _dos_setvect(DOS_INT, newint21);

    getch();
    printf("%lu syscalls\n", syscalls);

    /* alten Int21-Vektor wiederherstellen */

    _dos_setvect(DOS_INT, oldint21);
}
```

Listing 2: *Ein einfaches Programm, das sich in einen Interruptvektor einklinkt und dann zur Originalroutine weiterspringt.*

Interrupte treten auf, wenn eine Einheit einen Interrupt aktiviert. Jede Systemeinheit ist so programmiert, daß sie zu gewissen Zeiten Interrupte auslöst, und es müssen Routinen vorhanden sein, die diese Interrupte behandeln. Software-Interrupte treten auf, wenn der Befehl `INT` ausgeführt wird. In beiden Fällen wird die Steuerung an eine Interrupt-Behandlungsroutine übergeben, deren Adresse in einer Tabelle im unteren Speicherbereich des Computers gespeichert ist. Die Interrupt-Behandlungsroutine führt alles durch, was notwendig ist und führt dann den Befehl `IRET` aus, mit dem die Kontrolle an die Routine zurückgegeben wird, die beim Auftreten des Interrupts aktiv war.

Interrupt-Behandlungsroutinen für Hardware-Interrupte dürfen im allgemeinen den Zustand des Systems nicht verändern, d.h. keine Register verändern und auch nicht den Stack oder den Bildschirminhalt. Natürlich werden von Hardware-Interrupten einige Dinge schon verändert. Es gibt im Speicher einige besondere Variablen, die für die Kommunikation zwischen Interrupt-Behandlungsroutinen und normalen Programmen reserviert sind und die von der Interrupt-Routine verändert werden dürfen. Der Zeitgeber-Interrupt verändert zum Beispiel die Variable `ticks`. Die Interrupt-Routine dürfte jedoch nicht das AX-Register verändern, weil dadurch andere Programme fehlerhaft funktionieren würden.

Interrupt-Routinen für Software-Interrupte werden explizit aufgerufen und geben Ergebnisse meist in Registern zurück. So wird zum Beispiel das AX-Register von den meisten DOS-Funktionen verändert. Dies ist erlaubt und sogar notwendig. Wenn jedoch ein TSR-Programm einen Software-Interrupt abfängt, sollte es darauf achten, daß keine Register verändert werden.

In Assembler ist es relativ einfach, eine Routine zu entwickeln, die kein Register verändert. Man muß nur darauf achten, welche Register von der Routine verwendet werden und sie mit `PUSH` sichern und mit `POP` wiederherstellen. C-Routinen sichern die Register jedoch nicht so sorgfältig. In einer normalen Microsoft-C-Routine können die Register AX und DX zur Rückgabe von Werten verwendet werden, die Flags dürfen verändert werden, mit dem ES-Register kann man auf FAR-Daten zugreifen und so weiter. Es ist nur notwendig, daß eine Routine die Register SI,DI, DS, SS, BP und SP erhält. Dieses Verhalten ist möglich, weil der C-Compiler Code erzeugt, der sich an diese Regeln hält.

Microsoft C verfügt über das nur wenig bekannte und nicht besonders ausführlich dokumentierte Schlüsselwort `interrupt`. Wenn eine Funktion mit `interrupt` deklariert wird, werden mehr Register gerettet und der Rücksprung erfolgt mit `IRET` anstatt mit `RET`. Eine weitere Eigenschaft einer `interrupt`-Funktion ist, daß man Funktionsparameter deklarieren kann, die Registern entsprechen. Dadurch kann man die Registerwerte verwenden und den Registern auch neue Werte zuweisen.

In *Listing 1* sind eine Interrupt-Routine in C und der vom Compiler erzeugte Assemblercode zu sehen. Beachten Sie, daß Interrupt-Routinen immer als `far` deklariert werden müssen. Der Compiler weiß, daß eine Interrupt-Prozedur mit Sichern des Flagregisters, sowie von CS und IP aufgerufen wird. Bei der Beendigung werden diese von der `IRET`-Anweisung wieder vom Stack gelesen. Der Compiler weiß jedoch nicht, daß eine Interrupt-Prozedur ohne Parameter aufgerufen werden soll, selbst wenn 13 formale Parameter deklariert worden sind.

Bei der alten »Kernighan & Ritchie«-Deklaration in Listing 1 akzeptiert der Compiler beim Aufruf jede beliebige Anzahl Parameter. Wenn Sie stattdessen eine neue ANSI-Deklaration wie die folgende verwenden, erwartet der Compiler, daß sie genau 13 Parameter übergeben:

```
void interrupt far intfarfn(unsigned es,
unsigned ds, unsigned di, unsigned si,
unsigned bp, unsigned sp, unsigned bx,
unsigned dx, unsigned cx, unsigned ax,
unsigned ip, unsigned cs, unsigned flags)
{
}
```

In beiden Fällen werden Parameter, die sie versehentlich angeben, ignoriert. Es ist sehr wichtig, daß Sie die Parameterliste in der richtigen Reihenfolge angeben. Der C-Compiler erkennt die Parameter nicht am Namen, sondern an der Reihenfolge:

```
#include <dos.h>
extern unsigned int _psp, end;
extern unsigned char _osmajor;

main()
 {
     char huge *startofitall;
     char huge *endofitall;
     unsigned blength; /* Länge in Byte: psp+text+data */
     unsigned plength; /* Länge in Paragraphen */

     printf("DOS-Version %d\n", _osmajor);

     FP_SEG(startofitall) = _psp;
     FP_OFF(startofitall) = 0;
     endofitall = (char huge *)&end;
     blength = endofitall - startofitall; /* Bytes */
     plength = blength;
     if (plength & 0xf)  /* runden auf 16-Byte Para */
         plength += 0x10;
     plength >>= 4;      /* in Paragraphen umwandeln */

     printf("Start %Fp, Ende %Fp\n"
            "Größe %u (Byte) %u (Paragraphen)\n",
     startofitall, endofitall, blength, plength);
}
```

Listing 3: *Das Programm* `LOCS.C` *mit dem die Größe eines Programms festgestellt werden kann.*

Der erste Parameter ist ES, der zweite DS und so weiter. Wenn Sie zum Beispiel auf DI zugreifen wollen, der Inhalt von ES oder DS Ihnen aber gleich ist, können Sie die Funktion so deklarieren:

```
void interrupt far
intfarfn(unsigned junk1, junk2, di)
{
}
```

Die Parameter `junk1` und `junk2` werden benötigt, um `di` in der Parameterliste an der richtigen Stelle stehen zu haben.

In einen Interrupt-Vektor einklinken

Wenn ein TSR-Programm gestartet wird, ist es eine normale DOS-Task und hat die Kontrolle über das System. Es kann innerhalb der Grenzen von DOS tun und lassen, was es will. Wenn es jedoch die Steuerung an DOS zurückgegeben hat, kann es nur noch ausgeführt werden, wenn es durch einen Interrupt aktiviert wird. Deshalb muß jedes TSR-Programm vor der Beendigung einen Far-Pointer auf eine seiner Funktionen in einen Eintrag der Interrupt-Vektortabelle kopieren. Wenn dann der Interrupt auftritt, wird die angegebene Funktion aktiviert.

In den meisten Situationen ist es darüber hinaus erforderlich, daß die Funktion auch die ursprüngliche Interrupt-Routine für diesen Interrupt aufruft; dieser Vorgang wird »Einklinken« in einen Interrupt-Vektor genannt (engl. »hooking«).

```
DOS-Version 3
Start 2279:0000, Ende 2400:09B0
Größe 8736 (Byte) 546 (Paragraphen)
```

Bild 1: *Die Ausgabe von LOCS.*

Die meisten TSR-Programme klinken sich in mehrere Interrupte ein; sie können sich beispielsweise in den Tastaturinterrupt einklinken, um auf eine Aufruftaste zu warten, und in den DOS-Interrupt, um festzustellen, welche DOS-Funktionen gerade aktiv sind. Das Einklinken in Interrupte ist jedoch nicht auf TSR-Programme beschränkt, auch normale Programme können diese Möglichkeit nutzen. Kommunikationsprogramme klinken sich zum Beispiel häufig in die seriellen Dateninterrupte ein, und viele Programme verwenden den Interrupt für kritische Fehler und den Ctrl-Break-Interrupt.

Microsoft C enthält zwei Funktionen, die es vereinfachen, sich in einen Interrupt einzuklinken und zwar `_dos_getvect` und `_dos_setvect`. Die Funktion `_dos_getvect`, mit der ein Interruptvektor gelesen wird, wird zuerst verwendet. Ihr Parameter gibt an, welcher Vektor gelesen werden soll. Sie gibt einen Far-Pointer auf eine Funktion zurück, der gewöhnlich für die spätere Verwendung gespeichert wird.

Die Funktion `_dos_setvect` benötigt zwei Parameter: Der erste gibt den gewünschten Vektor an, und der zweite ist ein Far-Pointer auf eine Funktion, der in diesem Vektor gespeichert werden soll. Die Verwendung dieser beiden Funktionen wird in der Hauptroutine von *Listing 2* demonstriert. Beachten Sie bitte, daß dieses Programm kein TSR-Programm ist; es handelt sich hier um ein gewöhnliches Programm, das den DOS-Interrupt »ausspioniert«. Das Programm achtet darauf, daß der Vektor vor der Beendigung wiederhergestellt wird; wenn dies nicht gemacht würde, würde das System spätestens beim Laden der nächsten Anwendung abstürzen.

Die Deklaration einer Variablen, in der ein Far-Pointer gespeichert werden kann, ist nicht ganz einfach. Daß C eine solche Deklaration ermöglicht, ist eine ihrer Stärken, doch fast jedermann sagt, daß die Notation schlecht strukturiert ist. Die zwei Deklarationen zu Beginn von Listing 2 lauten:

```
void far interrupt newint21();
void (interrupt far * oldint21)();
```

In der ersten wird `newint21` als Far-Interruptfunktion ohne Rückgabewert deklariert. Mit dieser Deklaration wird kein Speicher belegt, es wird damit nur dem Compiler `newint21` bekannt gemacht, damit er den richtigen Code erzeugen kann. Die zweite Deklaration sieht ähnlich aus, ist jedoch gänzlich anders. Damit wird eine Variable mit dem Namen `oldint21` deklariert, die einen Far-Pointer auf eine Interruptfunktion enthält, die keinen Wert zurückgibt.

Beide Klammerpaare in der Deklaration von `oldint21` sind notwendig. Das erste bewirkt, daß sich `interrupt far *` auf `oldint21` bezieht, das zweite, leere Klammerpaar macht die Variable zu einem Zeiger auf eine Funktion. C-Deklarationen werden von innen nach außen gelesen: Das Sternchen wird als »Zeiger auf« gelesen, die Klammern als »Funktion, die etwas zuzrückgibt« und eckige Klammern als »Array aus«. Dinge, die rechts von einem Namen stehen (runde und eckige Klammern) binden stärker, als Dinge, die links stehen (* und Schlüsselwörter wie `far`, `near` und `interrupt`). Die Deklaration von `oldint21` wird so gelesen: `oldint21` ist ein Far-Interruptzeiger auf eine Funktion, die nichts (`void`) zurückgibt. Das ist genau das, was wir brauchen.

Wenn ein TSR-Programm sich in einen Interruptvektor einklinkt, gibt es drei Möglichkeiten zur Berücksichtigung der Original-Routine für diesen Vektor:

- die Original-Routine ignorieren,
- die Original-Routine aufrufen (`CALL`),
- zur Original-Routine weiterspringen (`JMP`).

TSR-Programme entscheiden sich oft erst aufgrund der aktuellen Situation, welche dieser Vorgehensweise sie verwenden. Eine Routine, die auf eine bestimmte Aufruftaste wartet, könnte beispielsweise zur Original-Routine weiterspringen (`JMP`), wenn eine Taste gedrückt wird, die nicht ihre Aufruftaste ist; sie würde die Original-Routine jedoch aufrufen (`CALL`) wenn die Aufruftaste festgestellt wird.

Die erste Vorgehensweise, das Ignorieren der Original-Routine ist nur in wenigen Situationen möglich. Normalerweise muß die Original-Routine aufgerufen werden, damit das System korrekt arbeiten kann. Die zweite Vorgehensweise, der Aufruf (`CALL`) der Interruptroutine, erlaubt es einem TSR-Programm, die Steuerung zu erhalten, nachdem die Original-Routine ihre Aufgabe erledigt hat. Dies ist nicht sehr häufig, doch es kommt vor.

Eine Funktion, deren Adresse in einem Funktions-Pointer wie `oldint21` gespeichert wurde, wird so aufgerufen:

```
(*oldint)();
```

Wieder sind beide Klammernpaare notwendig. Ohne das scheinbar überflüssige erste Klammerpaar würde `oldint21` wie eine Funktion aussehen anstatt wie Zeiger auf eine Funktion. Es ist enorm wichtig, daß `oldint21` als Far-Pointer auf eine Interruptfunktion und nicht als einfacher Far-Pointer deklariert wird, da bei Interruptfunktionen andere Aufrufkonventionen verwendet werden müssen.

Der dritte Weg, das Weiterspringen zur Original-Routine, ist sehr häufig. Die TSR-Routine führt einen `JMP` auf die Original-Routine aus. Vor dem `JMP` muß das TSR-Programm alle geänderten Register wiederherstellen und sicherstellen, daß der Stackzeiger und das Stack-Segmentregister wieder die Werte haben, die sie vor der Aktivierung des TSR-Programms hatten.

In einem Assemblerprogramm ist das Weiterspringen nicht schwierig, wenn man genau weiß, in welchem Zustand sich die Register befinden. In C muß dazu die Funktion `_chain_intr` verwendet werden, die jedoch nur aus einer Interrupt-Routine heraus aufgerufen werden darf, weil sie weiß, wie der Stack von einer Interruptroutine manipuliert wird und dieses Wissen dazu verwendet, den Stack zu korrigieren und dann zur angegebenen Far-Routine zu springen. Wenn die angesprungene Routine startet, sind die Registerinhalte und der Stackaufbau genauso, wie bei einem direkten Aufruf. Ein ganz einfaches Beispiel dafür ist die Funktion `newint21` in *Listing 2.*

Wenn das Programm aus Listing 2 ausgeführt wird, zeigt es normalerweise die Meldung »1 syscalls« an und meldet damit, daß genau ein Aufruf des DOS-Interrupts 21h erfolgt ist. Dabei handelt es sich natürlich um den Aufruf von `getch` in der Funktion selbst. Wenn Sie ein speicherresidentes Programm aufrufen, während dieses Beispielprogramm auf einen Tastendruck wartet, wird bei seiner Beendigung die Anzahl von dessen DOS-Aufrufen angezeigt.

Stackprobleme und ihre Lösung

Meistens brauchen sich C-Programmierer nicht um den Stack zu kümmern. Es ist hilfreich zu wissen, daß bei einer Hardware-Architektur wie der des PC C-Auto-Variablen auf dem Stack gespeichert werden. Eine gute allgemeine Regel bei der C-Programmierung lautet, in einer Funktion keine großen Datenstrukturen zu definieren. (Ein häufiger Fehler von Neulingen ist die Deklaration der wichtigen Datenstrukturen in der `main`-Routine.) Wenn der Stack zu klein ist, kompiliert man sein Programm neu und verlangt einen größeren Stack. Rekursive Programme benötigen meist einen größeren Stack als die voreingestellten 2 Kbyte.

Wenn ein gewöhnliches EXE-Programm abläuft, verwendet es seinen eigenen Stack, wenn jedoch eine DOS-Systemfunktion aufgerufen wird, schaltet DOS auf einen eigenen Stack um. Tatsächlich gibt es drei DOS-Stacks, einen für die Systemaufrufe 0 bis 0Ch, einen für die Funktionen über 0Ch und einen dritten für die Verwendung bei kritischen Fehlern. Ein TSR-Programm hat zwei Möglichkeiten: Es kann darauf vertrauen, daß der zur Zeit des Aufrufs aktive Stack genügend Platz hat, oder es kann auf einen eigenen Stack umschalten. Einige TSR-Programm schalten auf einen eigenen Stack um, nur um ganz sicher zu gehen. Es gibt zwei sehr wichtige Dinge, die Sie berücksichtigen sollten, wenn Sie den einfachen Weg gehen wollen, d.h. den aktuellen Stack verwenden.

Zum einen müssen Sie darauf achten, daß die Stack-Überprüfung bei Ihrem Programm ausgeschaltet ist, indem Sie die Compileroption `-Gs` verwenden, und daß Sie keine Bibliotheksfunktionen aufrufen, die eine Stacküberprüfung durchführen. Einige Bibliotheksfunktionen machen das, andere nicht. Verwenden Sie beim Kompilieren die Compileroption `-Fm` um eine MAP-Datei zu erzeugen und unter-

suchen Sie die Datei auf Verweise auf die Funktion chkstk. Wenn Sie solche Verweise finden, sollten Sie systematisch alle Bibliotheksfunktionen eliminieren, bis Sie chkstk aus Ihrem Programm verbannt haben.

Zum zweiten sollten Sie daran denken, bei der Verwendung des Stacks äußerst sparsam vorzugehen. Wenn Sie lokale Variablen mit static deklarieren, wird der Stack nicht mehr für lokale Variablen verwendet. Beachten Sie aber, daß lokale statische Variablen ihren Wert behalten und bei jedem Start der Routine neu initialisiert werden müssen; sie können sich nicht auf die automatische Initialisierung in einer Funktion verlassen, bis auf den ersten Aufruf. Um die Stackverwendung gering zu halten, sollten Sie keine Rekursion verwenden, nicht mehr als eine Handvoll Variablen auf dem Stack ablegen und Prozeduren nicht zu tief verschachteln. Wenn Ihr Programm sparsam mit dem Stack umgeht, brauchen Sie nicht auf einen anderen Stack umzuschalten.

Ein weiteres Stackproblem besteht darin, daß DS und SS bei der Ausführung der TSR-Routine eventuell nicht gleich sind. Sie sollten den Compilerschalter -Aw verwenden, damit der Compiler Code erzeugt, der nicht davon ausgeht, daß DS und SS den gleichen Wert haben. Wenn Sie -Aw verwenden, müssen Sie auch -Asn verwenden, um ein Small-Model-Programm zu erzeugen. Normalerweise erzeugt der Compiler keinen Code, der von DS==SS ausgeht, oft kommt man also auch ohne -Awsn aus. Wegen dieses Problems sollten Sie es vermeiden, die Adresse einer lokalen Auto-Variablen zu verwenden. Wie bereits erwähnt, sollten lokale Variablen in TSR-Programmen ganz vermieden werden, um die notwendige Stackgröße klein zu halten. Die Zurückhaltung bei der Verwendung ihrer Adressen erfordert also keine weiteren Einschränkungen.

Einige Bibliotheksroutinen gehen von DS==SS aus, doch der einzige Weg, diese herauszufinden, besteht im Probieren. Wenn eine Bibliotheksfunktion nicht richtig funktioniert, wenn sie in einem TSR-Programm verwendet wird, jedoch richtig, wenn sie in der gleichen Situation in einem normalen Programm eingesetzt wird, liegt das wahrscheinlich daran, daß sie von DS==SS ausgeht.

Bibliothekroutinen

Eine der schönsten Eigenschaften von Microsoft C ist seine Bibliothek. Microsoft C enthält eine Bibliothek mit umfangreichen Ein-/Ausgaberoutinen, einer vielseitigen mathematischen Bibliothek, Routinen für die Stringmanipulation, Funktionen für die dynamische Speicherverwaltung, eine allgemeine Betriebssystemschnittstelle und einige PC-spezifische Routinen. Mit der Version 5 kamen Grafikroutinen und weitere PC-spezifische Funktionen hinzu.

Der größte Teil der Microsoft C-Bibliothek kann nicht in einem TSR-Programm verwendet werden. Einige Routinen verursachen Probleme, weil sie direkt oder indirekt malloc aufrufen.

```
/*
 * TSR-Programm für die Zeitanzeige
 */

#include <dos.h>
#include <bios.h>

extern int _psp, end;
extern unsigned _osmajor;

union REGS regs;
int scr_row, scr_col;                /* Bildschirmzeile/-spalte */

int far *scr_cursaddr;
unsigned scr_ch;

void scr_swapmsg(unsigned char *, unsigned char *);
unsigned scr_read_ch();
void scr_write_ch();
int far * scr_get_curs_addr();

#define BIOS_VIDEOINT         0x10
#define BIOS_VIDEOINT_RDCH    0x8
#define BIOS_VIDEOINT_WRCH    0x9
#define BIOS_CURSOR_ARRAY     (int far *)(0x450)
#define BIOS_VIDEO_PAGE_NUM   (char far *)(0x462)


#define HOTKEY_CODE 0x11          /* Taste W */
#define HOTKEY_STAT 0x4           /* Ctrl */
#define KEYPORT     0x60          /* Tastatur-I/O-Port */


#define DOS_INT     0x21
#define KEYSVC_INT  0x9
#define DOS_PRINTSTRING 0x9

#define StartOfTime 6             /* Position 6 in msg */

char msg[25] = "****  HH:MM:SS  ****";
char buf[25];


/* Variablen für Zeitberechnungen */

unsigned long ticks;
int h,m,s;
char *sptr;



void (interrupt far * oldint9)();

/* Auf Aufruftaste warten und dann Zeit anzeigen mit popup() */

void interrupt far newint9(es, ds, di, si, bp, sp, bx, dx,
                           cx, ax, ip, cs, flags)
unsigned es, ds, di, si, bp, sp, bx, dx, cx, ax, ip, cs, flags;
{
    static unsigned char keycode;
    static char active = 0;

    if (active)
        _chain_intr(oldint9);

    keycode = inp(KEYPORT);
    if (keycode != HOTKEY_CODE)
    _chain_intr(oldint9);

    if (_bios_keybrd(_KEYBRD_SHIFTSTATUS) & HOTKEY_STAT)
    {
        /* Taste gedrückt */
        active = 1;
        (*oldint9)();   /* normale Verarbeitung der Taste */
        popup();
        active = 0;
    }
    else
        _chain_intr(oldint9);
}
```

```
main()
{
    char huge *startofitall;
    char huge *endofitall;
    unsigned length;

    /* DOS-Version überprüfen */
    if (_osmajor < 2)
    {
        dos_puts("Nicht geladen, DOS ab V2 notwendig.\r\n$");
        exit();
    }

    /* alten INT9 sichern */
    oldint9 = _dos_getvect(KEYSVC_INT);
    _dos_setvect(KEYSVC_INT,(void (interrupt far *)())newint9);

    /* Positions- und Längenberechnung */
    FP_SEG(startofitall) = _psp;
    FP_OFF(startofitall) = 0;
    endofitall = (char huge *)&end;
    length = endofitall - startofitall; /* Bytes */
    /* auf nächsten 16-Byte-Paragraph runden */
    if (length & 0xf)
        length += 0x10;
    length >>= 4;              /* in Paragraphen umwandeln */

    dos_puts("TSRTime geladen.\r\n$");
    /* Alles im Speicher lassen bis auf Stack */
    _dos_keep(0, length);
}


 /* Einen mit $ abgeschlossenen String anzeigen */

dos_puts(s)
char *s;
{
    regs.x.dx = (int)s;
    regs.h.ah = DOS_PRINTSTRING;
    intdos(&regs,&regs);
}


popup()
{
    static int oldcursor;

    /* Aufruftaste aus Puffer nehmen */
    (void)_bios_keybrd(_KEYBRD_READ);

    /* Cursor sichern */
    oldcursor = *scr_get_curs_addr();

    /* Zeit in die Meldung einbauen */
    bigben();

    /* Meldung anzeigen */
    scr_row = 12; scr_col = 30; scr_swapmsg(msg,buf);

    /* auf Tastendruck warten */
    (void)_bios_keybrd(_KEYBRD_READ);

    /* Meldung wieder löschen */
    scr_row = 12; scr_col = 30; scr_swapmsg(buf,msg);

    /* Cursor wiederherstellen */
    *scr_get_curs_addr() = oldcursor;
}


/*
 * ticks lesen, in Sekunden umwandeln
 * und in msg schreiben.
*/
bigben()
{
     _bios_timeofday(_TIME_GETCLOCK, &ticks);
    ticks *= 2000L;     /* umwandeln ...  */
    ticks /= 36413L;    /* ... in Sekunden */
    h = ticks / 3600;
    m = ticks / 60;
    m = m % 60;
    s = ticks % 60;
    sptr = msg + StartOfTime;
    *sptr++ = h/10 + '0';
    *sptr++ = h%10 + '0';
    *sptr++ = ':';
    *sptr++ = m/10 + '0';
    *sptr++ = m%10 + '0';
    *sptr++ = ':';
    *sptr++ = s/10 + '0';
    *sptr++ = s%10 + '0';
}

/*
 * Bildschirm-BIOS-Routinen für TSRs
 */
/*
 * Meldung auf dem Bildschirm ausgeben, vorher Bildschirm-
 * inhalt in savbuf sichern. scr_row und scr_col müssen
 * vor dem Aufruf von scr_swapmsg() belegt werden.
 */
void scr_swapmsg(msg, savbuf)
unsigned char *msg, *savbuf;
{
    static unsigned val;

    scr_cursaddr = scr_get_curs_addr();
    while(*msg)
    {
        *savbuf++ = scr_ch = scr_read_ch();
        scr_ch &= 0xff00;    /* Attribut erhalten */
        scr_ch |= *msg++;
        scr_write_ch();
        scr_col++;
    }
    *savbuf = 0;
}

/*
 * Zeichen vom Bildschirm in scr_ch lesen (scr_row, scr_col)
 */
unsigned scr_read_ch()
{
    /* Cursoradresse in BIOS-Variable schreiben */
    *scr_cursaddr = scr_row<<8 | scr_col;
    /* Zeichen über BIOS lesen */
    regs.h.ah = BIOS_VIDEOINT_RDCH;
    regs.h.bh = *BIOS_VIDEO_PAGE_NUM;
    int86(BIOS_VIDEOINT, &regs, &regs);
    return regs.x.ax;
}


/*
 * scr_ch in Bildschirm bei scr_row, scr_col schreiben.
 */
void scr_write_ch()
{
    /* Cursoradresse in BIOS-Variable schreiben */
    *scr_cursaddr = scr_row<<8 | scr_col;
    /* Zeichen über BIOS schreiben */
    regs.h.ah = BIOS_VIDEOINT_WRCH;
    regs.h.al = scr_ch;      /* Zeichen */
    regs.x.cx = 1;           /* ein Zeichen */
    regs.h.bh = *BIOS_VIDEO_PAGE_NUM;
    regs.h.bl = scr_ch>>8;   /* Attribut */
    int86(BIOS_VIDEOINT, &regs, &regs);
}

/*
 * Die Cursoradresse der aktuellen Seite feststellen
 */
int far * scr_get_curs_addr()
{
    return BIOS_CURSOR_ARRAY + (int)(*BIOS_VIDEO_PAGE_NUM);
}
```

Listing 4: *Ein TSR-Programm, das die Zeit anzeigt.*

Bei der Ausführung eines Programms verwaltet malloc einen Speicherbereich, und teilt ihn auf Anforderung auf. Doch dieser Speicher ist für ein TSR-Programm nicht verfügbar und jede Verwendung von malloc oder Varianten davon ist in einem TSR-Programm verboten. Einige Routinen beanspruchen den Stack zu stark, während andere Routinen chkstk aufrufen, was gewöhnlich zu einem Fehler führt, da TSR-Stacks sich normalerweise nicht dort befinden, wo chkstk sie erwartet.

Die Routinen, die auf jeden Fall sicher sind, sind diejenigen, die man einfach auch selbst schreiben kann. Eine Möglichkeit zur Lösung dieses Problems ist, den Quellcode für die Bibliotheksfunktionen zu erwerben, so daß man eine genauere Übersicht darüber hat, was in den einzelnen Funktionen vor sich geht. Wenn man den Quellcode besitzt, kann man benötigte Routinen auch so kompilieren, daß sie keine Stacküberprüfung mehr durchführen; die Microsoft-Routinen haben die Stacküberprüfung meist aus guten Gründen eingeschaltet, denn sie benötigen oft mehr als nur ein paar Bytes auf dem Stack. (Eine andere Möglichkeit zum Ausschalten der Stacküberprüfung besteht darin, das Modul CHKSTK.ASM anzupassen, das auf der Utilities-Diskette des C-Compilers mitgeliefert wird.)

Die Programmgröße

Obwohl Microsoft C mit _dos_keep über eine bequeme Unterroutine verfügt, mit der man ein aktives Programm in ein Hintergrundprogramm verwandeln kann, gibt es noch eine weitere Hürde zu überwinden. Man muß _dos_keep mitteilen, wie lang (in 16-Byte-Paragraphen) das Programm ist. Ein Weg, diesen Wert zu berechnen, besteht darin, zu schätzen, entweder indem man sich die MAP-Datei ansieht oder indem man einen größeren Wert angibt, als wirklich benötigt wird. Der bessere Weg besteht jedoch darin, es im Programm selbst auszurechnen.

MS-DOS-Programme haben den folgenden allgemeinen Aufbau:

- das Programmsegmentpräfix (PSP)
- Text (Programmcode)
- Daten
- Stack

Für unsere Zwecke ist die Größe des Stacks irrelevant, da er nicht benötigt wird, nachdem das Programm speicherresident beendet wird. Die Aufgabe besteht darin, die Größe der anderen Teile herauszufinden. Das Programmsegmentpräfix (PSP) ist einfach, es ist immer 256 Byte (100h) groß. Mit dem Programm LOCS.C in *Listing 3* habe ich eine Methode entwickelt, um die Größe von Text- und Datensegment zu bestimmen. Seine wichtigsten Bestandteile sind zwei Huge-Pointer, startofitall und endofitall. Ich belege startofitall so, daß er auf das PSP und endofitall so, daß er auf das Ende des Datensegments zeigt. Den Beginn des Programm kann man leicht herausfinden; die Variable _psp enthält immer die Segmentadresse des Beginns des PSP, und der Offset der Startadresse ist immer 0. Das Ende des Datensegments kann man mit Hilfe einer Variablen mit dem Namen end herausfinden, die immer das letzte Datenelement im Datensegment ist. Bei den Pointern muß es sich um Huge-Pointer handeln, da ihre Segmente verschieden sind. Durch subtrahieren der Huge-Pointer erhält man die Programmgröße. In *Bild 1* ist die Ausgabe des LOCS-Programms zu sehen. Um das Ergebnis von LOCS.C zu überprüfen, schaute ich in der MAP-Datei nach und kam zum selben Ergebnis.

Eine Feinheit von LOCS, auf die ich noch hinweisen möchte, ist die Umwandlung von Byte in Paragraphen. Vergessen Sie dabei nicht aufzurunden; 10 Byte ergeben zum Beispiel einen Paragraphen, 17 Byte sind zwei Paragraphen und so weiter. Die Anweisungen am Ende von LOCS runden auf (wenn nötig) und wandeln dann in Paragraphen um, indem um 4 nach rechts geshiftet wird (Division durch 16).

Beenden und resident bleiben

In *Listing 4* ist ein einfaches TSR-Programm zu sehen, das immer dann die Zeit anzeigt, wenn seine Aufruftaste gedrückt wird. Diese einfache Routine klinkt sich nur in den Tastaturinterrupt ein. Wenn die Aufruftaste gedrückt wird, wird die Zeit in der Bildschirmmitte angezeigt. In diesem einfachen TSR-Programm wird die ganze Arbeit mit BIOS-Aufrufen erledigt und es gibt keine Dateiein-/-ausgabe. Viele nützliche TSR-Programme sind so einfach, doch andere machen umfangreicheren Gebrauch von DOS.

Die main-Routine enthält eine Umsetzung der entsprechenden Programmteile aus den Listings 2 und 3. Zu Beginn der Routine wird der Vektor auf die alte Interrupt-9h-Routine gesichert und eine neue Interrupt-9h-Routine installiert. Dann folgt die Berechnungsroutine, mit der festgestellt wird, wieviel Platz das Programm benötigt, d.h. wieviel mit _dos_keep im Speicher gehalten werden soll. Darauf folgt der eigentliche Aufruf von _dos_keep, wodurch das Programm speicherresident wird. _dos_keep stellt nicht den gleichen Systemzustand her, wie dies eine normale Beendigung mit exit tun würde. Ein Unterschied ist natürlich, daß der Speicherbereich für das Programm und die Daten erhalten bleibt. Ein weiterer Unterschied besteht darin, daß _dos_keep mehrere Vektoren für Fließkommaoperationen, die vom C-Startcode installiert wurden, nicht wie exit wiederherstellt.

Das Einklinken in den Interrupt und das Weiterspringen erfolgen in der Funktion newint9. Bei jedem Tastaturinterrupt wird newint9 aktiviert. Wenn sicher ist, daß die Aufruftaste nicht betätigt wurde, wird zur Original-Interruptroutine weitergesprungen. Wenn jedoch die Aufruftaste festgestelllt wird, wird die Original-Routine mit CALL aufgerufen. Dadurch kann die Original-Routine Tastendrücke normal empfangen und verarbeiten. Sobald von der Original-Routine zum TSR-Programm zurückgekehrt wird, ruft

newint9 die Funktion popup auf, in der die Aufruftaste aus dem Tastaturpuffer gelesen, die Zeit angezeigt, auf eine weitere Tastaturbetätigung gewartet und dann der ursprüngliche Bildschirminhalt wiederhergestellt wird.

Die Funktion newint9 verwaltet ein Flag mit dem Namen active, das gesetzt wird, bevor popup aufgerufen und anschließend wieder zurückgesetzt wird. Dadurch verhindert man, daß newint9 popup ein zweites Mal aufruft, wenn die Aufruftaste gedrückt wird, während das TSR-Programm aktiv ist. Dies ist nicht weit hergeholt; es würde passieren, wenn der Benutzer die Aufruftaste einmal für die Zeitanzeige und ein zweites Mal zum Ausschalten der Anzeige drückt.

Das *Listing 4* enthält einige Bildschirm-Hilfsroutinen, die den Zeitstring anzeigen und hinterher den Bildschirm wiederherstellen. Diese Funktionen verwenden aus Gründen der Einfachheit und Allgemeingültigkeit das BIOS für die Ein-/Ausgabe. Für derart kurze Meldungen, wie in diesem Beispiel, ist die langsamere Geschwindigkeit des BIOS unwichtig.

Die hier verwendete Vorgehensweise funktioniert jedoch nicht korrekt mit Programmen, die die Cursorpositionierungsroutinen des BIOS umgehen, wie das zum Beispiel bei Lotus 1-2-3 der Fall ist. Die Routinen gehen davon aus, daß die Cursorposition mit der in den BIOS-Variablen gespeicherten identisch ist. Die einzige Schwierigkeit bei diesen BIOS-Routinen besteht darin, die Cursorposition zu erhalten. Dies wird dadurch erreicht, daß die Werte mit Hilfe der Funktion scr_get_curs_addr, die deren Speicheradresse berechnet, aus dem BIOS-Datenbereich ausgelesen werden.

Komplexe TSR-Programme

TSR-Programme, die Dateien lesen oder schreiben, Verzeichnisse auflisten oder andere Aufgaben ausführen, die DOS-Funktionen verwenden, müssen ausgefeilter sein, als das hier vorgestellte Programm. Bedenken Sie, daß DOS kein Multitasking-System ist. Wenn ein Aufruf erfolgt, schaltet DOS automatisch auf einen seiner internen Stacks um. Da ein normales Anwendungsprogramm inaktiv ist, während DOS eine Anforderung bearbeitet, ist es bei Nichtvorhandensein eines TSR-Programms unmöglich, daß ein weiterer Aufruf erfolgt, während der erste noch abgearbeitet wird. Wenn jedoch im Hintergrund TSR-Programme lauern, kann es passieren, daß ein Aufruf erfolgt, während ein anderer noch behandelt wird. Leider ist DOS so programmiert, daß es abstürzt oder anderweitig fehlerhaft reagiert, wenn sich Aufrufe überschneiden. Es ist nicht schwer, ein flexibleres Aufrufschema zu entwickeln, doch das schien damals, als DOS entwickelt wurde, nicht wichtig zu sein.

Wegen dieser Einschränkungen verwaltet DOS das sogenannte Busy-Flag. Ein Programm kann die Adresse dieses Flags feststellen, indem es den DOS-Systemaufruf 34h ausführt. Es wird ein Far-Pointer auf das Flag in ES:BX zurückgegeben. Während DOS einen Aufruf erledigt, setzt es das Busy-Flag. Das bedeutet nicht, daß DOS schlau genug ist, Probleme zu vermeiden, indem es signalisiert, daß es beschäftigt ist. Es bedeutet nur, daß es für ein TSR-Programm möglich ist, festzustellen, daß DOS aktiv ist und es so einen tödlichen Aufruf unterlassen kann.

Leider setzt DOS das Busy-Flag auch, wenn es eigentlich möglich wäre, die meisten DOS-Funktionen für Dateiein-/-ausgabe aufzurufen. Wenn DOS zum Beispiel auf eine Tastatureingabe wartet, sind Dateiein-/-ausgaben erlaubt. Ein Weg, diese Situation zu erkennen, besteht darin, sich in den Interrupt 21h einzuklinken um festzustellen, welche Funktion gerade ausgeführt wird. Das TSR-Programm könnte dann bei einem Systemaufruf feststellen, ob es wirklich sicher ist, Dateiein-/ausgaben auszuführen.

Die einfachste Lösung ist, DOS-Aufrufe zu unterlassen, wenn eine Funktion über 0Ch bearbeitet wird, und Funktionen über 0Ch zu erlauben, wenn eine Funktion unter 0Ch erledigt wird. Viele TSR-Programme verwenden auch den Status von Interrupt 13h, die BIOS-Diskettenein-/-ausgaberoutinen, um festzustellen, ob es möglich ist Dateiein-/-ausgaben durchzuführen.

Eine andere Möglichkeit besteht darin, eine Ressource von DOS zu verwenden, den Interrupt 28h. Wenn das Busy-Flag von DOS gesetzt ist, Funktionen über 0Ch aber möglich sind, ruft DOS regelmäßig den Interrupt 28h auf. Ursprünglich war der Interrupt 28h für den residenten Teil von PRINT.COM gedacht, es handelt sich dabei jedoch um einen allgemeingültigen Mechanismus, der von vielen TSR-Programmen verwendet wird. Einige TSR-Programme aktivieren den Interrupt 28h sogar selbst, wenn andere TSR-Programme zusätzlich zu ihnen aktiviert werden dürfen.

Viele TSR-Programme verwenden auch den regelmäßigen 18 Hz-Interrupt. Wenn ihre Aufruftaste gedrückt wurde, setzt die Routine, die in den Tastaturinterrupt eingeklinkt wurde, nur ein Flag, das anzeigt, daß die Aktivierung angefordert wurde. Kurz darauf erfolgt der Zeitgeber-Interrupt. Wenn DOS gerade nicht arbeitet, aktiviert die Routine, die in den Zeitgeber-Interrupt eingeklinkt wurde, die eigentliche TSR-Routine; sonst probiert es die Zeitgeber-Interruptroutine beim nächsten Mal. Dies wird häufig mit einer Routine kombiniert, die auf den Interrupt 28h reagiert. Die Routine, die zuerst auf die sichere Möglichkeit zur Aktivierung stößt, aktiviert die TSR-Routine und setzt das Flag für die Aktivierungsanforderung zurück.

Zwei weitere Dinge müssen berücksichtigt werden, zum einen der Tastaturinterrupt und der Interrupt für kritische Fehler. Es darf nicht möglich sein, ein TSR-Programm mit [Ctrl][Break] zu unterbrechen - eine Katastrophe wäre die sichere Folge. Auch sollte ein TSR-Programm den Interrupt 24h auf sich richten, wenn es Dateiein-/-ausgabe betreibt. Wenn es das nicht tut, könnte ein kritischer Fehler (die lästige Meldung »Abbrechen, Wiederholen, Ignorieren?«) zu unvorhersehbaren Ergebnissen führen. In beide

Interrupte sollte man sich nur einklinken, während ein TSR-Programm aktiv ist; die Vektoren sollten wiederhergestellt werden, wenn das TSR-Programm sich zurückzieht.

Eine weitere Schwierigkeit für viele TSR-Programme besteht darin, festzustellen, ob sie bereits geladen wurden. Es ist in der Regel nicht wünschenswert, ein TSR-Programm zweimal zu laden, doch die versteckte »Hinter-den-Kulissen-Natur« von TSR-Programmen verführt Benutzer oft dazu, sie ein zweites Mal zu laden, wenn nach dem ersten Laden Schwierigkeiten auftreten.

Es gibt für ein TSR-Programm mehrere Möglichkeiten, festzustellen, ob es bereits geladen wurde. Ein einfacher Weg besteht darin, den ganzen Speicher (vom Anfang bis zum Beginn des PSP) nach etwas Eindeutigem wie einer Copyright-Meldung zu durchsuchen. Wenn der String gefunden wird, muß das TSR-Programm schon geladen worden sein. Eine andere Möglichkeit ist die Verwendung von Interrupten; man kann beispielsweise einem existierenden Interrupt eine neue Funktion hinzufügen. Vor der speicherresidenten Installation ruft das TSR-Programm den Interrupt mit diesem Funktionscode auf. Wenn das TSR-Programm bereits geladen ist, übergibt es einen speziellen Code. Wenn die normale Interrupt-Routine aufgerufen wird, wird die übliche Routine ausgeführt, daß heißt, in der Regel passiert nichts.

Zusammenfassung

Welche Sprache Sie auch verwenden, die Schwierigkeiten beim Schreiben von TSR-Programmen resultieren hauptsächlich aus den Unzulänglichkeiten von DOS und liegen nicht an der Sprache selbst. Assemblerprogrammierer haben hier die Nase vorn, weil sie für TSR-Programme die gleichen Tricks und Techniken verwenden können, die sie immer verwenden.

Viele der Microsoft C-Fähigkeiten zum Schreiben von TSR-Programmen sind ungewohnt, weil sie für andere Programmierweisen fast gar nicht benötigt werden. Das Erstellen eines TSR-Programms ist jedoch relativ einfach, wenn man die Microsoft C-Tools und die grundlegenden Tricks der TSR-Programmierung verstanden hat.

Zusammenfassend kann man sagen: Für einfache TSR-Programme kann man eine beliebige Sprache verwenden, C oder Assembler, je nachdem, mit welcher man vertrauter ist. Für viele Programmierer ist das C. Für ganz anspruchsvolle TSR-Programme ist jedoch immer noch Assembler die bevorzugte Sprache.

Kaare Christian

Ein nützliches TSR-Programm in Assembler:

Bildschirminhalte auf Knopfdruck speichern

Es wird zwar relativ viel geschrieben über TSR-Programme, doch meistens sind die Beispiele nur sehr einfach und drücken sich vor allem um den schwierigsten Teil, den Aufruf von DOS und die Probleme damit, daß DOS nicht reentrant ist. In diesem Artikel wird ein Programm vorgestellt, daß auch diese Klippen umschifft.

Doch bevor ich die Einzelheiten der Programmierung einer solchen TSR-Utility beschreibe, möchte ich zunächst das Programm aus Benutzersicht erläutern.

Bild - Elektronische Bildschirmfotografie

Bild ist ein Programm, mit dem Textbildschirminhalte von PC auf Diskette gespeichert werden können. Das ist sehr nützlich, wenn man PC-Programme beschreiben muß und auch Bildschirmabbildungen in seinen Text einfügen möchte. Die Speicherung erfolgt so, wie die Daten im Bildschirmspeicher stehen, mit dem Programm *BldTxt* können sie jedoch in Standard-ASCII-Dateien umgewandelt werden. Mit eigenen Utilities wäre auch die Umsetzung in das Format der verwendeten Textverarbeitung denkbar.

Bild wird nach einmaligem Aufruf fest im Speicher installiert und kann dann jederzeit mit einer Tastenkombination aktiviert werden. Diese Tastenkombination kann mit dem Installationsprogramm *BildInst* vom Benutzer bestimmt werden.

Bild belegt etwa 5 Kbyte Speicher. Es unterstützt folgende Bildschirmmodi:

2 80 x 25 schwarz-weiß (MODE BW80)
3 80 x 25 Farbe (MODE CO80)
7 80 x 25 monochrom (MODE MONO)

Bild wird durch Eingabe von `BILD` [Return] gestartet, und meldet sich dann mit einem Copyright-Hinweis (*Bild 1*). Das Programm kann nun jederzeit durch Betätigung der beim Installationsprogramm *BildInst* (s.u.) angegebenen Tastenkombination (Standard: [Alt][Return]) gestartet werden. Bei der Betätigung der installierten Tastenkombination wird der derzeitige Bildschirminhalt in einer Datei gespeichert. Die Textbildschirmdateien erhalten die Namen `BILDxxx.BLD`, wobei `xxx` eine Zahl zwischen 001 und 999 ist. Wenn eine Datei bereits vorhanden ist, überschreibt *Bild* sie nicht, sondern zählt so lange weiter, bis es eine Nummer findet, für die noch keine Datei existiert.

Die Dateien werden immer in das aktuelle Verzeichnis geschrieben. Wenn also eine Anwendung »fotografiert« wird, die häufig die Verzeichnisse wechselt, können sich die Bildschirmdateien in verschiedenen Verzeichnissen befinden. Befindet sich der Bildschirm in einem Modus, der von Bild nicht unterstützt wird (z.B. Grafikmodus), piepst das Programm. Ebenso bei einem kritischen Fehler (z.B. Diskette defekt oder nicht eingelegt).



Bild 1: *Die Startmeldung von* Bild.

Bild kann beliebig oft aufgerufen werden. Es belegt dann keinen weiteren Speicher, sondern meldet nur, daß es bereits geladen wurde. Der Startbildschirm wird nur beim ersten Aufruf angezeigt. Beim Aufruf kann der Parameter `A` angegeben werden. *Bild* wird dann ausgeschaltet. Der von dem Programm verwendete Speicher ist verloren (ca. 5 Kbyte). Es braucht kein Schrägstrich oder Bindestrich vor dem Parameter angegeben werden.

Wenn *Bild* bereits geladen wurde und erneut aufgerufen wird, wird das mit folgender Meldung angezeigt:

`BILD: Programm bereits installiert.`

Wird *Bild* mit dem Parameter `A` (ausschalten) aufgerufen, wird das mit dieser Meldung bestätigt:

`BILD: Programm ausgeschaltet.`

Wird *Bild* mit dem Parameter `A` (ausschalten) aufgerufen, ist es aber noch gar nicht geladen worden, so wird folgende Meldung ausgegeben:

`BILD: Programm nicht installiert, kann nicht ausgeschaltet werden.`

Bild wird dann nicht aktiviert, sondern muß erneut ohne den Parameter `A` aufgerufen werden.

Bild benötigt eine DOS-Version ab 2.11 (besser ab 3.0). Sollte die DOS-Version älter sein, erscheint diese Meldung:

`BILD: DOS 2.11 oder neuer erforderlich.`

Bild wird bei dieser Meldung nicht installiert.

Installationsprogramm

Bevor *Bild* verwendet werden kann, sollte es mit dem Programm *BildInst* installiert werden. Für die Installation muß sich die Datei `BILD.COM` im aktuellen Verzeichnis befinden. Das Programm wird durch Eingabe von `BILDINST` [Return] aufgerufen. *BildInst* fragt folgende Einstellungen ab (*Bild 2*):

- Ob das Programm das Speichern der Bildschirme hörbar hinweisen soll.
- Mit welcher Tastenkombination *Bild* aktiviert werden kann. Hier können die auf dem Bildschirm angegebenen Tastennamen verwendet werden.

Eine Besonderheit sind die Tasten mit vorangestelltem *#*. Dabei handelt es sich um die entsprechenden Tasten im Zehnerblock. Des weiteren können die beiden Shift-Tasten durch `RShift` und `LShift` unterschieden werden.

Nachdem alle Eingaben gemacht worden sind, fragt *BildInst*, ob die Werte so installiert werden sollen.

```
BILD Installation 1.00 (C) G.Jürgensmeier 10/88
Einstellungen: Ton EIN   Aufruf mit: Alt Return

Mit Ton? (J/N/Esc): J

Shift-Tasten:
Normal RShift LShift Ctrl Alt

Tasten:
A-Z      ÄÖÜ      0-9       <ß'#     +,.-     Esc      Backspace Tab
Return   PrtSc    Leertaste Caps     F1       F2       F3        F4
F5       F6       F7        F8       F9       F10      NumLock   ScrollLock
Home     Oben     PgUp      #-       Links    #5       Rechts    #+
End      Unten    PgDn      Ins      Del      SysReq   F11       F12

Neue Tastenkombination? ([Return]=keine Änderung)

Gewählte Taste: Alt Return
Installieren mit [Return], abbrechen mit [Esc], [W] = Wiederholen.
```

Bild 2: BildInst, *das Installationsprogramm für* Bild.

An dieser Stelle besteht die Möglichkeit, das Programm durch Eingabe von [Esc] zu beenden. Mit [Return] werden die Werte im Programm *Bild* gespeichert. Mit [W] ist eine Wiederholung der Eingabe einer Tastenkombination möglich.

Utilities

`BLDTXT`

Mit dem Programm *BLDTXT* können BLD-Dateien in Standard-ASCII-Dateien umgewandelt werden. Die Ein- und Ausgabedatei werden abgefragt. Die Dateinamenerweiterungen brauchen nicht angegeben werden.

`ZEIGEBLD`

Mit *ZeigeBLD* können BLD-Dateien angezeigt werden. Es kann immer nur eine Datei angegeben werden, z.B. `ZEIGEBLD PCCODES` (die Erweiterung braucht nicht angegeben werden).

Grundlagen der TSR-Programmierung

Bevor ich auf die Einzelheiten des Programms *Bild* eingehe, möchte ich zunächst die Grundlagen von speicherresidenten Programmen beschreiben. Die DOS-Funktionen Terminate and Stay Resident (Interrupt 21h, Funktion 31h und Interrupt 27h) ermöglichen es, ein Programm oder Daten im Speicher fest zu installieren. Programme, die diese Möglichkeiten nutzen, werden deshalb kurz TSR-Programme genannt.

TSR-Programme lassen sich in zwei Kategorien einordnen, die einfachen verwenden keine DOS-Aufrufe und die komplizierteren benutzen wie ein normales Programm die DOS-Funktionen, besonders die für die Datei-Ein- und Ausgabe.

Die Programme der ersten Gruppe sind relativ leicht zu schreiben, sogar in höheren Programmiersprachen, wie im vorhergehenden Artikel gezeigt wurde. Sie warten still im Hintergrund, bis ihre Aufruftaste betätigt wird, erledigen dann einfache Aufgaben (wie z.B. die Anzeige von Zeit und/oder Datum) und sind deshalb ziemlich unproblematisch.

```
          name     BILD
          page     65,132
          title    Textbildschirme speichern

;=================================================================
;
; BILD:            Textbildschirme speichern
;         (C)      G. Jürgensmeier,  1986-88
;                  Version 3.00 vom 25.10.88
;
;=================================================================
CodeSeg Segment
        assume cs:CodeSeg, ds:CodeSeg

CR        equ      0dh
LF        equ      0ah

;=================================================================
                   org      40h       ;
Dta                db       64 dup (?); DTA
                   org      80h       ;
Parameter          db       128 dup (?);
                   org      100h      ; für COM-Datei
StackTop           equ      $         ; lokaler Stack, Obergrenze

;=================================================================
Start:                                ;
          jmp      Resident           ; Programm resident machen

;=================================================================
Marker             dw       'LB'      ; BL-Kennzeichen für TestIns
                   db       " GJ "    ; Autor
                   db       "10/88"   ; Version
HotKey             db       28        ; Aufruftaste, Standard: Return
HotShift           db       8         ; Shift-Status, Standard: Alt
Ton                db       1         ; Ton ein

;=================================================================
TextSeg            dw       0b000h    ; wird geändert, falls nötig
TextLen            dw       80*25*2   ;
                                      ;
TextHandle         dw       0         ; File-Handle BLD-Datei
TextName           db       'BILD'    ; Dateiname
TextNummer         db       '001'     ; Nummer
                   db       '.BLD',0;
                                      ;
PrgAktiv           db       0         ; Programm läuft
BiosDiskFlag       db       0         ; Status von INT 13h
SpeicherFlag       db       0         ; Speicherflag
FehlerFlag         db       0         ; kritischer Fehler INT 24h
TonFlag            db       0         ; Flag für summen
BreakFlag          db       0         ; DOS-Einstellung Break

InDOS              dd       ?         ; Adresse InDOS-Flag

TastaturInt        dd       ?         ; alter INT 09h Vektor
TimerInt           dd       ?         ; alter INT 1Ch Vektor
BiosDiskInt        dd       ?         ; alter INT 13h Vektor
DosIdleInt         dd       ?         ; alter INT 28h Vektor
CritErrInt         dd       ?         ; alter INT 24h Vektor
DtaAdr             dd       ?         ; alte DTA-Adresse

SaveSP             dw       0         ; Sicherung Stapelzeiger
SaveSS             dw       0         ; Sicherung Stapelsegment

;=================================================================
Tastatur Proc      Near               ; bei jedem Tastendruck
         assume    ds:Nothing         ;
         sti                          ; Interrupts erlaubt
         push      ax                 ; AX wird gebraucht
         in        al,60h             ; Scan-Code lesen
         cmp       al,cs:HotKey       ; war es die Aufruftaste?
         jne       Tastatur2          ; nein
```

Listing 1: *Das Programm* Bild.

```
        mov     ah,2              ; ja, Shift-Status prüfen
        int     16h               ;
        and     al,0fh            ; nur unteren 4 Bit
        cmp     al,cs:HotShift    ; war es die Shift-Kombination?
        jne     Tastatur2         ; nein
        call    TastaturReset     ; ja, Taste ignorieren
        pop     ax                ;
                                  ;
        cmp     PrgAktiv,0        ; Programm bereits aktiv?
        jne     Tastatur1         ; ja, Flag nicht setzen
        mov     SpeicherFlag,1    ; Flag setzen
                                  ;
Tastatur1:                        ;
        iret                      ;
                                  ;
Tastatur2:                        ;
        pop     ax                ;
        jmp     TastaturInt       ; alte Tastaturroutine aufrufen
Tastatur EndP

;==============================================================
Timer   Proc    Near              ; 18mal pro Sekunde
        assume  ds:Nothing        ;
        pushf                     ; alte Routine aufrufen
        call    TimerInt          ;
                                  ;
        cmp     SpeicherFlag,0    ; Flag gesetzt?
        je      Timer1            ; nein, beenden
                                  ;
        push    es                ; ES & DI werden gebraucht
        push    di                ;
        les     di,InDOS          ; Zeiger auf InDOS-Flag
        cmp     Byte Ptr es:[di],0; ist DOS aktiv?
        pop     di                ;
        pop     es                ;
        jne     Timer1            ; ja, Programm muß warten
                                  ; DOS nicht aktiv
        cmp     BiosDiskFlag,0    ; BIOS aktiv?
        jne     Timer1            ; ja, nicht unterbrechen
                                  ; BIOS nicht aktiv
        mov     al,20h            ; EOI an 8295 PIC
        out     20h,al            ;
                                  ;
        mov     SpeicherFlag,0    ; Flag zurücksetzen
                                  ;
        call    SaveScreen        ; Hauptprogramm aufrufen
                                  ;
Timer1: iret                      ;
Timer   EndP

;==============================================================
BiosDisk Proc   Near              ; INT 13h Behandlung
        assume  ds:Nothing        ;
        inc     BiosDiskFlag      ; Flag erhöhen
        pushf                     ; Original-Routine aufrufen
        call    BiosDiskInt       ;
        dec     BiosDiskFlag      ; Flag wieder vermindern
        iret                      ;
BiosDisk EndP

;==============================================================
DosIdle Proc    Near              ; INT 28h Behandlung
        assume  ds:Nothing        ;
        pushf                     ; Original-Routine aufrufen
        call    DosIdleInt        ;
        cmp     SpeicherFlag,0    ; Flag gesetzt?
        je      DosIdle1          ; nein, ignorieren
                                  ;
        mov     SpeicherFlag,0    ; Flag zurücksetzen
                                  ;
        call    SaveScreen        ; Hauptprogramm aufrufen
                                  ;
DosIdle1:                         ;
        iret                      ;
DosIdle EndP
```

Listing 1 *(Fortsetzung)*

Viele einfache TSR-Programme kümmern sich einfach nicht um den Zustand des Systems oder anderer gerade aktive Programme. Oft gehen sie davon aus, daß der Benutzer weiß, wann er das Programm aufrufen kann und wann nicht. In vielen Fällen ist diese Vorgehensweise durch den Nutzen des Programms gerechtfertigt, doch einen ehrgeizigen Programmierer können solche Lösungen nicht befriedigen. Er strebt nach sicheren TSR-Programmen, die auch funktionieren, wenn gerade andere Programme kritische Aufgaben ausführen.

Das sind dann TSR-Programme der zweiten Kategorie. Das bekannteste Beispiel dafür ist SideKick. SideKick macht viele Dinge, die für ein speicherresidentes Programm offiziell eigentlich gar nicht möglich sind, zum Beispiel auf Dateien zugreifen.

Solange man nicht besondere Vorsichtsmaßnahmen ergreift, können die DOS-Funktionen, die über die Interrupte 21h, 25h und 26h aufgerufen werden, in TSR-Programmen nicht verwendet werden. Die Verwendung einer solchen Funktion führt fast sicher zum Systemabsturz. Programme wie SideKick sind jedoch der Beweis dafür, daß es eine Möglichkeit geben muß. Es gibt sie wirklich und die Tricks, die es ermöglichen, unterscheiden die einfachen von den ausgefeilten TSR-Programmen.

Bevor man sich daran macht, TSR-Programme zu schreiben, sollte man verstehen, warum DOS-Funktionen von einem normalen Programm aus beliebig aufgerufen werden können, von einem TSR-Programm jedoch nicht. Der Grund ist, daß die DOS-Funktionen nicht reentrant sind. Das bedeutet, daß DOS-Funktionen nicht aufgerufen werden können, wenn bereits eine aktiv ist. Normalerweise hat ein Programmierer auch nicht die Möglichkeit, eine DOS-Funktion aufzurufen, während eine andere bearbeitet wird, den die aufgerufene Funktion gibt die Kontrolle erst nach vollständiger Verarbeitung an das aufrufende Programm zurück. Ein speicherresidentes Programm kann jedoch jederzeit aufgerufen werden und so auch eine aktive DOS-Funktion unterbrechen. Das speicherresidente Programm wird in diesem Fall wahrscheinlich noch laufen, es kommt jedoch fast unweigerlich zur Katastrophe, wenn die Kontrolle an die unterbrochene Funktion zurückgegeben wird und diese versucht, die begonnene Arbeit fortzusetzen. Ein Systemabsturz ist sicher und man kann von Glück reden, wenn dabei keine Daten verloren gehen.

Man sollte auch wissen, warum DOS nicht reentrant ist. Wenn der Interrupt 21h ausgeführt wird, besteht eine der ersten Dinge, die DOS erledigt, darin, auf einen der drei Stacks umzuschalten, die es intern verwaltet. Das gleiche gilt für die Interrupte 25h und 26h. Umgeschaltet wird dadurch, daß die Inhalte der Register SS und SP zuerst gesichert und dann mit neuen Werten geladen werden. Sie zeigen dann auf einen reservierten Datenbereich innerhalb von DOS. Wenn eine DOS-Funktion eine andere aufruft, die den gleichen Stack verwendet, werden alle Werte auf dem Stack (z.B. die Registerwerte beim Aufruf) vom zwei-

ten überschrieben. Überschrieben wird auch die Rücksprungadresse des ersten Aufrufs. Wenn also der erste Funktionsaufruf einen IRET-Befehl ausführt, wird zu einer unvorhersehbaren Stelle im Speicher gesprungen.

Welcher der drei internen Stacks benutzt wird, hängt vom Interrupt bzw. beim Interrupt 21h von der jeweiligen Funktion ab. Die Interrupte 25h und 26h verwenden immer den gleichen Stack. Die Funktionen 1 bis 0Ch von Interrupt 21h und einige Funktionen mit höheren Nummern verwenden einen anderen Stack. Die Funktionen 0 sowie 0Dh und höher (mit einigen Ausnahmen) verwenden den gleichen Stack wie die Interrupte 25h und 26h.

Warum Programme wie DOS interne Stacks verwenden, ist eine Frage der Programmierphilosophie. Andere Routinen im PC, wie zum Beispiel die BIOS-Routinen, sind in gewissem Maße reentrant, weil sie keinen eigenen Stack verwenden. Sie gehen davon aus, daß das Programm, das sie aufruft, genug Platz auf dem Stack für das PUSHen einiger Werte aufweist. Man kann so gut wie immer davon ausgehen.

Viele speicherresidente Programme verwenden ihren eigenen Stack, damit sie absolut sicher sind, daß ihre Stackerfordernisse das System nicht überfordern und dann das System abstürzen lassen. Die Verwendung des aktuellen Stacks ist jedoch einfacher, erfordert weniger Programmcode und funktioniert auch meistens. Ein eigener Stack wird deshalb oft nicht verwendet, es ist jedoch immer ratsamer, einen zu verwenden.

DOS-Zugriffe

Wenn man also den Nachteil umgehen will, daß DOS nicht reentrant ist, muß man sicherstellen, daß die eigene Routine nur dann läuft, wenn keiner der Interrupte 21h, 25h und 26h aktiv ist. Eine Methode, dies sicherzustellen, besteht darin, jeden dieser Interrupte abzufangen, ein Flag zu setzen, wenn einer der Interrupte aufgerufen wird und es zurückzusetzen, wenn davon zurückgekehrt wird. Es gibt jedoch eine einfachere Methode. DOS selbst verwaltet nämlich schon ein Flag, das jedes Programm abfragen kann, um zu sehen, ob gerade eine DOS-Routine aktiv ist. Dieses Flag wird das InDOS-Flag genannt. Die Adresse dieses Flags erhält man durch Aufruf von Interrupt 21h, Funktion 34h. Die Segementadresse wird in ES übergeben, der Offset in BX. Wenn man dieses Flag vor der Aktivierung eines speicherresidenten Programms abfragt, kann man sicherstellen, daß keine Systemfunktion unterbrochen wird.

Wenn Sie diese DOS-Funktion im DOS-Handbuch oder anderen Nachschlagewerken nachlesen wollen, werden Sie meistens nur den Hinweis finden: »Wird von DOS intern verwendet«. Diese Funktion ist bis heute noch nicht offiziell dokumentiert, obwohl sie bereits seit der Version 2 existiert. Doch viele Programmierer haben DOS genauer untersucht und vor allem PRINT disassembliert, das diese Funktion und andere nicht dokumentierte DOS-Fähigkeiten nutzt.

```
;==================================================================
CritErr Proc     Near              ; INT 24h Behandlung
        assume   ds:Nothing        ;
        sti                        ; Interrupts erlaubt
        mov      FehlerFlag,1      ; nur Fehler-Flag setzen
        mov      al,0              ; Fehler ignorieren
        iret                       ;
CritErr EndP

;==================================================================
SaveScreen Proc Near               ; Hauptprogramm
        assume   ds:Nothing        ;
        mov      PrgAktiv,1        ; Flag für Programm aktiv
        push     ax                ; AX auf alten Stack
        mov      cs:SaveSS,ss      ; Stapelzeiger sichern
        mov      cs:SaveSP,sp

        cli                        ;
        mov      ax,offset StackTop ; lokalen Stack einrichten
        mov      sp,ax             ;
        mov      ax,cs             ;
        mov      ss,ax             ;
        sti                        ; Interrupts erlaubt

        push     bx                ;
        push     cx                ;
        push     dx                ;
        push     si                ;
        push     di                ;
        push     ds                ;
        push     es                ;
                                   ;
        push     cs                ; DS = CS
        pop      ds                ;
        assume   ds:CodeSeg        ;
        push     cs                ; ES = CS
        pop      es                ;
                                   ;
        mov      ah,15             ;
        int      10h               ; Modus feststellen
        cmp      al,2              ; 2, 3 & 7 sind Ok
        je       SaveBLD           ;
        cmp      al,3              ;
        je       SaveBLD           ;
        cmp      al,7              ;
        je       SaveBLD           ;
DoBeep:                            ;
        call     Beep              ; Fehlermeldung
                                   ;
Exit:                              ;
        mov      PrgAktiv,0        ; Programm nicht mehr aktiv
        pop      es                ; Register wiederherstellen
        pop      ds                ;
        pop      di                ;
        pop      si                ;
        pop      dx                ;
        pop      cx                ;
        pop      bx                ;
        cli                        ; alten Stack wieder einrichten
        mov      ax,cs:SaveSP      ;
        mov      sp,ax             ;
        mov      ax,cs:SaveSS      ;
        mov      ss,ax             ;
        sti                        ;
        pop      ax                ; vom alten Stack
        ret                        ;

;==================================================================
SaveBLD:                           ; Modus noch in AL
        cmp      al,7              ; monochrome Karte?
        jnz      Color             ; nein
        mov      TextSeg,0b000h    ;
        jmp      Short TextOpen    ;
Color:                             ;
        mov      TextSeg,0b800h    ; Farb/Grafik-Adapter
```

Listing 1 *(Fortsetzung)*

```
TextOpen:                                ;
        mov     FehlerFlag,0             ; Fehlerflag zurücksetzen
        call    Prepare                  ; DTA & INT 24h einstellen
        mov     ax,02000h                ; summen
        call    TonEin                   ;
OpenText:                                ; BLD-Datei öffnen
        mov     dx,Offset TextName; testen, ob vorhanden
        mov     ax,3d00h                 ; zum Lesen öffnen
        int     21h                      ;
        jc      TextErr                  ; wenn Fehler
        cmp     FehlerFlag,0             ; krit. Fehler aufgetreten?
        jne     TextError                ; ja
                                         ;
        mov     bx,ax                    ; Handle in bx
        mov     ah,3eh                   ; diese Datei schließen
        int     21h                      ;
        call    IncTextName              ; nächste Datei probieren
        jmp     OpenText                 ;
                                         ;
TextErr:                                 ;
        cmp     ax,2                     ; Datei nicht gefunden?
        je      TextDo                   ; ja, kann geöffnet werden
        jmp     ErrorExit                ; beenden mit Hinweis
                                         ;
TextDo:                                  ;
        mov     dx,Offset TextName;
        mov     cx,0020h                 ; Attribut archive
        mov     ax,3c00h                 ; Datei anlegen (CREAT)
        int     21h                      ;
        mov     TextHandle,ax            ;
        jnc     TextCrit                 ; kein Fehler
        jmp     TextClose                ; beenden
TextCrit:                                ;
        cmp     FehlerFlag,0             ; krit. Fehler aufgetreten?
        jz      TextOk                   ; nein, Ok
TextError:                               ; kritischer Fehler
        call    Beep                     ; piepsen
        jmp     ErrorExit                ; noch einmal piepsen & beenden
                                         ;
TextOk:                                  ;
        push    ds                       ;
        push    es                       ;
        push    si                       ;
        push    di                       ;
                                         ;
        push    cs                       ; Bildschirminhalt kopieren
        pop     es                       ; ES = CS
        mov     ax,TextSeg               ; aus dem Bildschirmspeicher
        mov     ds,ax                    ; ds:si = b000:0
                                         ;
        mov     ah,15                    ; Bildschirm-Page
        int     10h                      ;
        mov     al,bh                    ; page
        xor     ah,ah                    ;
        mov     dx,4096                  ;
        mul     dx                       ; Größe
        mov     si,ax                    ; vom Bildschirmpuffer
                                         ;
        mov     si,ax                    ; vom Bildschirmpuffer
        mov     di,Offset Buffer; in internen Puffer
        mov     cx,es:TextLen            ; kopieren
        cld                              ;
  rep   movsb                            ;
                                         ;
        pop     di                       ;
        pop     si                       ;
        pop     es                       ;
        pop     ds                       ;
                                         ;
                                         ;
        mov     bx,TextHandle            ;
        mov     cx,TextLen               ; 2000 Zeichen
        mov     dx,Offset Buffer; Adresse der Bildschirmdaten
        mov     ah,40h                   ; in die BLD-Datei schreiben
        int     21h                      ;
        jc      WriteErrT                ;
        cmp     FehlerFlag,0             ; Fehler aufgetreten?
        jnz     WriteErrT                ; ja
```

Listing 1 *(Fortsetzung)*

Es gibt jedoch trotzdem noch ein Problem. Wie kann man ein speicherresidentes Programm aufrufen, wenn eine DOS-Funktion wie 0Ah aktiv ist, die eine ganze Eingabezeile liest, bevor zurückgekehrt wird? Das ist nämlich beispielsweise immer der Fall, wenn DOS auf eine Eingabe wartet. Hier kommt ein ebenfalls undokumentierter DOS-Interrupt zur Hilfe, der für die TSR-Programmierung unentbehrlich ist, der Interrupt 28h. Wenn die Funktion 0Ah auf Eingaben wartet, ruft Sie laufend den Interrupt 28h auf. Dieser Interrupt kann von speicherresidenten Programmen abgefangen werden. Wenn der Interrupt 28h ausgeführt wird, können DOS-Funktion größer als 0Ch ohne Probleme aufgerufen werden, selbst wenn das InDOS-Flag gesetzt ist. Zu diesem Zeitpunkt wird nämlich ein anderer Stack verwendet, als von diesen Funktionen, wie oben schon erläutert wurde.

Wenn man die DOS-Funktion 34h und den Interrupt 28h geschickt nutzt, lassen sich TSR-Programme schreiben, die gefahrlos die Dateifunktionen von DOS verwenden können. Es gibt natürlich noch einige Einschränkungen, so kann man zum Beispiel ein nach diesen Regeln geschriebenes Programm nicht aktivieren, wenn eine lange Datei mit `TYPE` angezeigt wird. Dann ist nämlich ständig das InDOS-Flag gesetzt. Doch damit kann man wohl leben.

Rücksicht auf andere

Zusätzlich zu den Rücksichten, die DOS erfordert, muß ein TSR-Programm auch darauf achten, daß es nicht in Konflikt mit einem anderen gerade aktiven Programm gerät. Anwendungsprogramme (normale und auch residente) benötigen zum Beispiel einen Speicherbereich, über den Daten von der Diskette gelesen werden, die DTA (Disk Transfer Address). Ein speicherresidentes Programm muß eine eigene DTA verwenden, die alte Adresse sichern und vor der Beendigung seiner Arbeiten wiederherstellen.

Berücksichtigt werden müssen auch kritische Fehler. Bei einem kritischen Fehler (z.B. einer fehlerhaften Diskette) erzeugt DOS den Interrupt 24h. Dieser Interrupt zeigt normalerweise auf die Routine, die die vielgeliebte Meldung »A(bbruch), W(iederholen), I(gnorieren?« anzeigt. Jedes Programm sollte diesen Interrupt abfangen und eine eigene Fehlerbehandlungsroutine vorsehen. Das gilt in besonderem Maße für ein TSR-Programm, das bei Problemen mit an Sicherheit grenzender Wahrscheinlichkeit auch anderen Programmen Ärger macht.

Bild im Detail

Wie so oft reicht es auch bei TSR-Programmen nicht aus, daß man die Theorie kennt. Erst wenn man ein Beispiel gesehen hat, weiß man wirklich, wie es geht. Dazu dient das Programm *Bild* (*Listing 1*).

Wenn *Bild* aufgerufen wird, wird zunächst zum Label `Resident` gesprungen (im letzen Drittel des Listings). Der

Programmcode ab dieser Stelle wird nur für die Initialisierung benötigt und steht deshalb am Ende des Programms. Er wird beim Residentmachen nämlich nicht mit übernommen, da er für die eigentliche Funktion von *Bild* nicht mehr benötigt wird. Als erstes überprüft *Bild* die DOS-Version und stellt dann die Adresse des InDOS-Flags fest (`InFlag`). Diese Adresse sollte immer im nichtresidenten Teil des Programms abgefragt werden, da es bei der späteren Abfrage Konflikte mit DOS geben könnte. Dann wertet es die Parameterzeile aus, um festzustellen, ob der Parameter `A` angegeben wurde. Der Programmteil ab dem Label `PrgEin` wird dann ausgeführt, wenn der A-Parameter nicht angegeben wurde. Hier werden nun folgende Interrupte abgefangen und deren Adressen gesichert):

9	Tastatur-Interrupt (`Tastatur`)
1Ch	Timer-Interrupt (`Timer`)
13h	BIOS-Disk-Interrupt (`BiosDisk`)
28h	Interrupt DOS wartet (`DosIdle`)

(Der Interrupt 24h wird hier noch nicht umgeleitet, dies erfolgt erst kurz bevor wirklich Dateizugriffe gemacht werden.) Darauf wird eine Copyright-Meldung ausgegeben (Label `Meldung`) und das Programm mit der DOS-Funktion 31h speicherresident gemacht. Diese Funktion benötigt in DX die Länge des Programms in Paragraphen von je 16 Byte. Dieser Wert wird hier berechnet (wobei ein Puffer für die Bildschimrdaten berücksichtigt wird).

Die Befehle ab dem Label `PrgAus` werden ausgeführt, wenn *Bild* mit dem Parameter `A` (ausschalten) aufgerufen wurde. Hier werden dann die gesicherten Interruptvektoren aus dem speicherresidenten *Bild*-Programm herausgelesen und wiederhergestellt. Auch hier wird dem Benutzer die Aktion mit einer Meldung angezeigt.

Die Programmteile `PrgEin` und `PrgAus` rufen beide als erstes die Routine `TestIns` auf, die feststellt, ob *Bild* bereits geladen wurde oder nicht. Das kann festgestellt werden, indem einer der »verbogenen« Interrupte gelesen und nachgesehen wird, ob sich hinter dem ersten JMP-Befehl des eventuell geladenen Programms der String "BL" befindet. Dies ist die Kennung des *Bild*-Programms (Label `Marker` am Anfang). Wenn das Programm bereits installiert ist, wird das Carry-Flag gesetzt. Die Programmzeilen bei dem Label `NichtIns` werden von `PrgAus` aufgerufen, wenn *Bild* mit dem Parameter `A` gestartet wurde, aber noch nicht installiert ist. Die Routine `SchonIns` wird ausgeführt, wenn *Bild* zum zweitenmal aufgerufen wird, ohne zwischendurch ausgeschaltet worden zu sein. Doch schauen wir uns nun den Kern des Programms an.

Der residente Teil

Am Anfang des Programms, gleich hinter dem Sprung zum nichtresidenten Teil des Programms, stehen einige Variablen, die mehrere Zwecke erfüllen. Die Variable `Marker` ist ein String, der die Programmkennung ("BL", umgedreht,

```
        cmp     ax,TextLen          ; angegebene Länge geschrieben?
        je      TextClose           ; ja, alles Ok

;=============================================================
WriteErrT:                          ;
        mov     bx,TextHandle       ;
        mov     ah,3eh              ; Datei schließen
        int     21h                 ;
        mov     dx,Offset TextName;
        mov     ax,4100h            ; Datei wieder löschen
        int     21h                 ;
ErrorExit:                          ;
        call    Beep                ;
SaveExit:                           ;
        call    Reset               ;
        jmp     Exit                ;

;=============================================================
TextClose:                          ;
        mov     bx,TextHandle       ;
        mov     ah,3eh              ; Datei schließen
        int     21h                 ;
        call    IncTextName         ;
        call    TonAus              ;
        jmp     SaveExit            ;

SaveScreen EndP

;=============================================================
IncTextName Proc Near               ;
        mov     bx,Offset TextNummer+2; ASCII-Zahl
IncText:                            ;
        inc     Byte Ptr [bx]       ;
        cmp     Byte Ptr [bx],'9';
        jle     IncTEnd             ;
        mov     Byte Ptr [bx],'0';
        dec     bx                  ;
        cmp     bx,Offset TextNummer;
        jl      IncTEnd             ;
        jmp     IncText             ;
IncTEnd:                            ;
        ret                         ;
IncTextName EndP

;=============================================================
Beep    Proc    Near                ; piepsen
        cmp     cs:TonFlag,1        ;
        jne     BeepDo              ;
        call    TonAus              ;
BeepDo:                             ;
        mov     al,0b6h             ;
        out     43h,al              ;
        mov     ax,1000h            ;
        out     42h,al              ;
        mov     al,ah               ;
        out     42h,al              ;
        in      al,61h              ;
        mov     ah,al               ;
        or      al,3                ;
        out     61h,al              ;
        sub     cx,cx               ;
        mov     bl,1                ;
BLoop:  loop    BLoop               ;
        dec     bl                  ;
        jnz     BLoop               ;
        mov     al,ah               ;
        out     61h,al              ;
        ret                         ;
Beep    EndP

;=============================================================
TonEin  Proc    Near                ; beim Speichern summen
        cmp     cs:Ton,1            ; Tonhöhe in AX
        jne     TonRet              ;
```

Listing 1 *(Fortsetzung)*

```
        mov     cs:TonFlag,1        ; Flag setzen
                                    ;
        push    ax                  ;
        mov     al,0b6h             ;
        out     43h,al              ;
        pop     ax                  ;
        out     42h,al              ;
        mov     al,ah               ;
        out     42h,al              ;
                                    ;
        in      al,61h              ;
        or      al,3                ; ein
        out     61h,al              ;
        ret                         ;
TonEin  EndP

TonAus Proc     Near                ; Summen beenden
        cmp     cs:Ton,1            ;
        jne     TonRet              ;
        cmp     cs:TonFlag,1        ;
        jne     TonRet              ;
                                    ;
        in      al,61h              ;
        and     al,NOT 3            ; aus
        out     61h,al              ;
        mov     cs:TonFlag,0        ; Flag zurücksetzen
TonRet:                             ;
        ret                         ;
TonAus EndP

;==============================================================
TastaturReset Proc Near             ; EOI an 8259 PIC
        in      al,61h              ; aktuellen Wert lesen
        mov     ah,al               ; in AH zwischenspeichern
        or      al,80h              ; Bit setzen
        out     61h,al              ; an Port ausgeben
        mov     al,ah               ; Originalwert
        out     61h,al              ; wiederherstellen
        cli                         ; Interrupts nicht erlaubt
        mov     al,20h              ; EOI-Wert
        out     20h,al              ; an 8259
        sti                         ; Interrupts wieder erlaubt
        ret
TastaturReset EndP

;==============================================================
GetKey  Proc    Near                ; Tastendruck lesen
        mov     ah,1                ; auf Taste prüfen
        int     16h                 ;
        jne     GetKey1             ; Taste gedrückt
                                    ; keine Taste:
        int     28h                 ; INT 28h aufrufen
        jmp     GetKey              ;
GetKey1:                            ;
        mov     ah,0                ; Taste lesen
        int     16h                 ;
        ret                         ;
GetKey  EndP


;==============================================================
Prepare Proc    Near                ; DTA & INT 24h einstellen
        push    es                  ; ES wird gebraucht
        mov     ah,2fh              ; DTA lesen
        int     21h                 ;
        mov     Word Ptr DtaAdr,es; sichern
        mov     Word Ptr DtaAdr+2,bx;
        mov     ah,1ah              ; neue DTA einstellen
        mov     dx,Offset Dta       ;
        int     21h                 ;
                                    ;
        mov     ah,35h              ; INT 24h Vektor lesen
        mov     al,24h              ;
        int     21h                 ;
```

Listing 1 *(Fortsetzung)*

weil Intel-Wort), die Initialen des Programmautors und die Programmversion (das Erstellungsdatum) enthält. Die Programmkennung wird, wie bereits beschrieben, dazu verwendet, um zu erkennen, ob *Bild* bereits installiert wurde. Sie wird des weiteren vom später vorgestellten Installationsprogramm *BildInst* verwendet, um festzustellen, ob das richtige Programm gepatcht wird. Auch der Autor und die Programmversion werden von *BildInst* überprüft, damit nichts schiefgehen kann und nicht eventuell ein falsches Programm gepatcht wird.

Hinter diesem String stehen drei weitere wichtige Variablen: `HotKey`, `HotShift` und `Ton`. Bei `HotKey` und `HotShift` handelt es sich um die Tastenkombination, mit der *Bild* aufgerufen werden kann. Sie steht hier am Anfang, damit sie vom Programm *BildInst* leicht angepaßt werden kann. Die möglichen Werte von `HotKey` und `HotShift` werden bei der Beschreibung von *BildInst* ausführlich erläutert. Die Variable `Ton` entscheidet darüber, ob das Speichern eines Bildschirms von einen Summton begleitet wird oder nicht. Auch dieser Wert kann mit *BildInst* angepaßt werden.

Der Tastaturinterrupt wurde von der Initialisierungsroutine so umgeleitet, daß er die Routine `Tastatur` ausführt. Hier wird ein Zeichen von der Tastatur gelesen und überprüft, ob es sich dabei um die Aufruftaste (`HotKey`/`HotShift`) handelt. Ist dies nicht der Fall, wird einfach zum Original-Tastaturinterrupt weitergesprungen. Wurde die Aufruftaste gedrückt, wird sofort die Tastatur zurückgesetzt, damit dieser Tastencode nicht an ein anderes Programm weitergeleitet wird. Dann wird das Flag `PrgAktiv` geprüft. Dieses Flag ist gesetzt, wenn *Bild* gerade dabei ist, einen Bildschirm zu speichern. Wenn *Bild* in diesem Moment erneut aufgerufen wird (wahrscheinlich durch zufälliges zweimaliges Betätigen der Aufruftaste), darf der Speichervorgang nicht aktiviert werden. Wenn alles in Ordnung ist, setzt die Routine `Tastatur` dann das Flag `SpeicherFlag`, das anderen Routinen angibt, daß der Bildschirm gespeichert werden soll. Der eigentliche Speichervorgang erfolgt nicht im Tastaturinterrupt, sondern im Timerinterrupt oder im Interrupt 28h.

Der Timerinterrupt ruft 18 mal pro Sekunde die Routine `Timer` auf. Hier wird zunächst die Original-Timerroutine aufgerufen. Dann wird überprüft, ob das Anforderungsflag zum Speichern des Bildschirms gesetzt ist. Wenn nicht, wird der Interrupt ohne weitere Aktion beendet. Wurde jedoch die Speicherung des Bildschirms angefordert, überprüft die Routine, ob DOS oder BIOS aktiv sind. Ist das nicht der Fall, kann die eigentliche Bildspeicherroutine `SaveScreen` aufgerufen werden. Zuvor wird jedoch das Anforderungsflag `SpeicherFlag` zurückgesetzt, damit ein Bildschirm nicht mehrfach gespeichert wird.

Die Befehle bei dem Label `BiosDisk` verfolgen, ob der BIOS-Disk-Interrupt aktiv ist. Dies wird dadurch erreicht, daß das Flag `BiosDiskFlag` vor jedem Aufruf erhöht und danach wieder vermindert wird.

Die Routine DosIdle wird bei jedem Interrupt 28h aufgerufen. Dies ist dann der Fall, wenn DOS auf Eingaben wartet (s.o.). Auch hier wird wieder zunächst die Original-Routine aktiviert. Wenn nun das Anforderungsflag gesetzt ist, wird SaveScreen aufgerufen. In diesem Fall braucht das InDOS-Flag nicht überprüft werden, da der Interrupt 28h ja angibt, daß DOS-Funktionen über 0Ch problemlos aufgerufen werden können.

Die Routine CritErr wird beim Speichern von Bildschirmen dazu verwendet, den Interrupt 24h für kritische Fehler abzufangen. Hier wird dann nur in einem Flag gespeichert, daß ein Fehler aufgetreten ist und DOS angewiesen, den Fehler zu ignorieren. Die Fehlerabfrage und -behandlung ist dann Sache des Programms selbst.

Hilfsroutinen

Bevor ich nun das Hauptprogramm erläutere, möchte ich erst noch einige Hilfsroutinen beschreiben, die weiter hinten im Programmlisting stehen. Mit der Routine TastaturReset wird die Tastatur zurückgesetzt und damit die Aufruftaste von der Verarbeitung durch eventuell gerade aktive Programme ausgeschlossen. Die Routine GetKey wird im Programm *Bild* nicht benötigt, sie steht hier nur als Beispiel dafür, wie man sicherstellen kann, daß auch weitere TSR-Programme noch aufgerufen werden können, wenn ein anderes aktiv ist und Tastatureingaben durchführt. Dann sollte man zusammen mit dem Interrupt 16h zur Tastaturabfrage regelmäßig den Interrupt 28h aufrufen, wenn keine Taste gedrückt wurde. Dadurch wird sichergestellt, daß auch ein anderes TSR-Programm noch aktiviert werden kann.

Die Routinen Prepare und Reset dienen dazu, den Systemzustand vor der Aktivierung des TSR-Programms zu sichern bzw. hinterher wiederherzustellen. Bei Prepare wird die DTA gesichert, der Interrupt 24h (kritischer Fehler umgeleitet) und die DOS-Break-Einstellung ausgeschaltet. Die DTA wird bei der Ausführung des residenten Teils auf einen Bereich im Programmsegmentpräfix eingestellt, da dieses dann nicht mehr benötigt wird. Reset stellt alle Werte bei Beendigung des TSR-Programms wieder auf die Ausgangswerte zurück.

Das Hauptprogramm

Das Hauptprogramm SaveScreen selbst muß zunächst noch einige Verwaltungsaufgaben erledigen, bevor der Bildschirm gespeichert werden kann. Zunächst wird das Flag PrgAktiv gesetzt, das angibt, das die Bildschirmspeicherung erfolgt und dann ein lokaler Stack eingerichtet, damit der gerade aktive Stack nicht belastet wird. Dann werden alle Register gesichert. Wenn dies geschehen ist, hat das TSR-Programm die Kontrolle über das System und kann innerhalb der aufgezeigten Grenzen tun und lassen, was es will.

```
          mov     Word Ptr CritErrInt,es; sichern
          mov     Word Ptr CritErrInt+2,bx;
          mov     ah,25h           ; neuen Vektor einstellen
          lea     dx,CritErr       ;
          int     21h              ;
          pop     es               ; ES wiederherstellen
                                   ;
          mov     ah,33h           ; DOS-Einstellung Break lesen
          mov     al,0             ;
          int     21h              ;
          mov     BreakFlag,dl     ; Einstellung merken
          mov     ah,33h           ;
          mov     al,1             ; Break-Überprüfung ausschalten
          mov     dl,0             ;
          int     21h              ;
          ret                      ;
Prepare EndP

;=================================================================
Reset     Proc    Near             ; DTA & INT24h wiederherstellen
          mov     ah,25h           ; INT 24h Vektor
          mov     al,24h           ;
          mov     dx,Word Ptr CritErrInt;
          push    ds               ; DOS braucht Seg in DS
          assume  ds:Nothing       ;
          mov     ds,Word Ptr CritErrInt+2;
          int     21h              ;
                                   ;
          mov     ah,1ah           ; DTA
          mov     dx,Word Ptr DtaAdr;
          mov     ds,Word Ptr DtaAdr+2;
          int     21h              ;
          pop     ds               ;
          assume  ds:CodeSeg       ;
                                   ;
          mov     ah,33h           ; DOS-Flag für Break einstellen
          mov     al,1             ;
          mov     dl,BreakFlag     ;
          int     21h              ;
          ret                      ;
Reset     EndP


;=================================================================
Buffer    Label   Word             ; Start des Puffers
                                   ;
Resident Proc     Near             ; Programm reident machen
          assume  ds:CodeSeg       ; Parameter auswerten
          mov     ah,30h           ; DOS-Version feststellen
          int     21h              ;
          cmp     al,0             ; zu alt
          jnz     InFlag           ;
                                   ;
          mov     dx,Offset MelDos; falsche DOS-Version
          mov     al,1             ; Status, nicht resident
          jmp     Meldung          ;
                                   ;
InFlag:                            ;
          push    es               ;
          mov     ah,34h           ; Adresse des InDOS-Flags
          int     21h              ; lesen
          mov     Word Ptr InDOS,bx;
          mov     Word Ptr InDOS+2,es; sichern
          pop     es               ;
                                   ;
          cld                      ;
          mov     si,Offset Parameter;
                                   ;
ParSchleife:                       ;
          lodsb                    ;
          cmp     al,CR            ; Parameterende?
          je      PrgEin           ;
                                   ;
ParamA:                            ;
          cmp     al,'a'           ; ausschalten
          je      PrgAusDo         ;
```

Listing 1 *(Fortsetzung)*

```
        cmp     al,'A'          ;
        jne     ParSchleife     ;
PrgAusDo:                       ;
        jmp     PrgAus          ;

;==============================================================
PrgEin:                         ;
        call    TestIns         ; bereits installiert?
        jnc     PrgEinInts      ;
        jmp     SchonIns        ; ja
                                ;
PrgEinInts:                     ;
        push    es              ;
        mov     ah,35h          ; INT-Vektoren einstellen
        mov     al,9h           ; Tastatur
        int     21h             ;
        mov     Word Ptr TastaturInt,bx;
        mov     Word Ptr TastaturInt+2,es;
        mov     ah,25h          ;
        lea     dx,Tastatur     ;
        int     21h             ;
                                ;
        mov     ah,35h          ; Timer
        mov     al,1ch          ;
        int     21h             ;
        mov     Word Ptr TimerInt,bx;
        mov     Word Ptr TimerInt+2,es;
        mov     ah,25h          ;
        lea     dx,Timer        ;
        int     21h             ;
                                ;
        mov     ah,35h          ; BIOS Disk
        mov     al,13h          ;
        int     21h             ;
        mov     Word Ptr BiosDiskInt,bx;
        mov     Word Ptr BiosDiskInt+2,es;
        mov     ah,25h          ;
        lea     dx,BiosDisk     ;
        int     21h             ;
                                ;
        mov     ah,35h          ; DOS
        mov     al,28h          ;
        int     21h             ;
        mov     Word Ptr DosIdleInt,bx;
        mov     Word Ptr DosIdleInt+2,es;
        mov     ah,25h          ;
        lea     dx,DosIdle      ;
        int     21h             ;
                                ;
        pop     es              ;
                                ;
        mov     dx,Offset Hinweis;
        mov     al,0            ; Status, resident
        jmp     Meldung         ;

;==============================================================
PrgAus:                         ; Original INTs wiederherst.
        call    TestIns         ; Programm bereits installiert?
        jc      PrgRemove       ; ja
        jmp     NichtIns        ; nein
                                ;
PrgRemove:                      ;
        push    ds              ;
        push    es              ;
                                ;
        mov     ah,35h          ; Int-Vektor lesen
        mov     al,9h           ; Tastatur
        int     21h             ; ES = Segment von INT 9
        mov     ax,Word Ptr es:TastaturInt+2; Vektor lesen
        mov     ds,ax           ;
        mov     dx,Word Ptr es:TastaturInt;
        mov     ah,25h          ; Vektor wiederherstellen
        mov     al,9h           ;
        int     21h             ;
                                ;
```

Listing 1 *(Fortsetzung)*

Hier wird zunächst festgestellt, ob ein gültiger Bildschirmmodus eingestellt ist, da *Bild* nur Textbildschirme speichert. Ist die nicht der Fall (und bei anderen Fehlern), wird gepiepst und mit der Routine `Exit` weitergemacht. Hier wird das Aktivflag wieder zurückgesetzt, die Register wiederhergestellt, der alte Stack wieder aktiviert und zur aufrufenden Routine (`Timer` oder `DosIdle`) zurückgekehrt.

Die eigentliche Speicherung des Bildschirms macht nur einen geringen Teil des Programms aus. Als erstes wird festgestellt, an welcher Adresse der Bildschirmspeicher steht. Das Fehlerflag von `CritErr` wird zurückgesetzt, der Systemzustand mit `Prepare` gesichert und für das TSR-Programm eingestellt. Damit der Benutzer merkt, daß der Bildschirm gespeichert wird, wird der Ton eingeschaltet.

Nun wird geprüft, ob eine Datei mit der aktuellen Nummer bereits existiert. Wenn dies der Fall ist, wird die Zahl so lange hochgezählt (mit der Unterroutine `IncTextName`), bis eine Nummer gefunden wurde, für die noch keine Datei existiert. Diese Datei wird dann zum Schreiben geöffnet.

Da es bei einigen Bildschirmkarten Probleme gibt, wenn man direkt aus dem Bildschirmspeicher auf Diskette schreibt, wird der Bidlschirmspeicherinhalt zunächst in einen internen Puffer (`Buffer`) kopiert. Dieser Puffer wird dann als unbearbeitetes Speicherbabbild in die Datei geschrieben. Falls beim Schreiben ein Fehler auftritt, Wird `WriteErrT` ausgeführt, sonst `TextClose`. Von `TextClose` wird die Datei geschlossen, der Ton wieder ausgeschaltet und die Systemparameter mit `Reset` wieder auf die Ausgangswerte zurückgesetzt. Im Falle eines Fehler sieht die Vearbeitung ähnlich aus, die geöffnete Datei wird jedoch wieder gelöscht und zusätzlich ein Piepser ausgegeben (`Beep`).

Das war's dann auch schon. Der Bildschirm ist gespeichert und kann weiterverarbeitet werden. Die QuickBASIC-*Listings 3 und 4* zeigen Beispiel, wie auf die Dateien zugegriffen werden kann. Die Programme *ZeigeBld* und *BldTxt* wurden oben schon beschrieben.

Das Installationsprogramm

Es bleibt nun nur noch das Installationsprogramm *BildInst* zu beschreiben. Es ist mit QuickBASIC 4.0 erstellt worden, worüber Sie sich vielleicht wundern. Ich wundere mich nur immer wieder über die Programmierer, die über BASIC die Nase rümpfen. Durch solche Dialekte wie QuickBASIC ist die Sprache BASIC schon lange mit anderen Sprachen gleichgezogen und man kann damit genauso strukturiert programmieren. Ich pflege für jede Programmieraufgabe die dafür angemessene Sprache einzusetzen und das sind meines Erachtens: Assembler für Systemprogramme oder Routinen, die schnell sein müssen; BASIC für nicht so wichtige Installationsprogramme und Entwicklungswerkzeuge; C für Alles, was bleibenden Bestand hat.

Wert	RShift	LShift	Ctrl	Alt
0				
1	■			
2		■		
3	■	■		
4			■	
5	■		■	
6		■	■	
7	■	■	■	
8				■
9	■			■
10		■		■
11	■	■		■
12			■	■
13	■		■	■
14		■	■	■
15	■	■	■	■

Tabelle 1: *Von der Funktion 2 des Interrupts 16h wird in den unteren vier Bit von AL verschlüsselt angegeben, welche Shift-Taste gerade gedrückt ist.*

Doch zurück zum Installationsprogramm *BildInst* (*Listing 2*). Die wichtigsten Bestandteile dieses Programms sind die Daten am Ende und die beiden Routinen `Decode` und `Encode`. In den DATA-Zeilen beim Label `ShiftMasks` sind die Shift-Tasten gespeichert. Da die Shift-Tasten in der Variable `HotShift` und im Programm *Bild* über Bits verschlüsselt sind, muß der Wert vor dem Patchen erst errechnet werden (*Tabelle 1*). In den DATA-Zeilen hinter dem Label `KeyCodes` sind die Namen der Tasten an der für sie gültigen Nummernposition abgelegt. Mit der Routine `Encode` wird eine im Klartext eingegeben Tastenkombination (z.B. `LShift RShift Esc`) in die entsprechenden Werte für `HotKey` und `HotShift` umgewandelt. Mit der Routine `Decode` können diese Werte umgekehrt wieder in Klartext umgewandelt werden. Die eingegebenen und umgerechneten Tasten werden dann von *BildInst* in das Programm `BILD.COM` gepatcht, nachdem anhand der in der Datei gespeicherten Informationen sichergestellt ist, daß es sich dabei um eine Programmdatei handelt, die von *BildInst* gepatcht werden darf.

Fazit

Wie sie sehen, ist die Programmierung eines TSR-Programms in Assembler relativ einfach, wenn man einmal gesehen hat, wie es geht. Ich denke, Sie werden nun in der Lage sein, auf der Basis der hier vorgestellten Programme ihr eigenes Super-TSR-Programm zu erstellen. Wenn Sie dabei ein Erfolgsprogramm wie SideKick entwickeln sollten, lassen Sie es mich wissen.

Günter Jürgensmeier

```
        mov     ah,35h          ; Int-Vektor lesen
        mov     al,1ch          ; Timer
        int     21h             ; ES = Segment von INT 1c
        mov     ax,Word Ptr es:TimerInt+2; Vektor lesen
        mov     ds,ax           ;
        mov     dx,Word Ptr es:TimerInt;
        mov     ah,25h          ; Vektor wiederherstellen
        mov     al,1ch          ;
        int     21h             ;
                                ;
        mov     ah,35h          ; Int-Vektor lesen
        mov     al,13h          ; BIOS Disk
        int     21h             ; ES = Segment von INT 13
        mov     ax,Word Ptr es:BiosDiskInt+2; Vektor lesen
        mov     ds,ax           ;
        mov     dx,Word Ptr es:BiosDiskInt;
        mov     ah,25h          ; Vektor wiederherstellen
        mov     al,13h          ;
        int     21h             ;

                                ;
        mov     ah,35h          ; Int-Vektor lesen
        mov     al,28h          ; DOS
        int     21h             ; ES = Segment von INT 28
        mov     ax,Word Ptr es:DosIdleInt+2 ; Vektor lesen
        mov     ds,ax           ;
        mov     dx,Word Ptr es:DosIdleInt ;
        mov     ah,25h          ; Vektor wiederherstellen
        mov     al,28h          ;
        int     21h             ;
                                ;
        pop     es              ;
        pop     ds              ;
                                ;
        mov     dx,Offset MelAus;
        mov     al,1            ; Status, nicht resident
        jmp     Meldung         ;
                                ;
NichtIns:
        mov     dx,Offset MelNein;
        mov     al,2            ; Status,nicht resident,Fehler
        jmp     Meldung         ;


;=============================================================
SchonIns:                       ; bereits installiert
        mov     dx,Offset MelPar;
        mov     al,2            ; Status, nicht resident
        jmp     Meldung         ;


;=============================================================
Meldung:                        ;
        push    ax              ; Status sichern
        mov     ah,9            ; String anzeigen
        int     21h             ;
        pop     ax              ;
        cmp     al,0            ; wenn Fehler
        jne     NichtRes        ; nicht resident machen
                                ; bis zum Label 'Resident'
        mov     dx,Offset Resident ; im Speicher stehen lassen
        add     dx,TextLen      ; Bildschirmpuffer Text
        shr     dx,1            ; /2
        shr     dx,1            ; /4
        shr     dx,1            ; /8
        shr     dx,1            ; /16
        inc     dx              ; aufrunden
        mov     ah,31h          ; keep
        int     21h             ;
                                ;
NichtRes:                       ;
        sub     al,1            ; Errorlevel setzen
        mov     ah,4ch          ; Beenden, Errorlevel in AL
        int     21h             ;
Resident EndP
```

Listing 1 *(Fortsetzung)*

```
;=============================================================
; TestIns:
;       Prüfen, ob Programm bereits installiert,
;       wenn ja, Carry setzen.
;       Test über Marker: muß 'LB' sein, wenn installiert
;=============================================================
TestIns Proc    Near            ;
        push    es              ;
        mov     ah,35h          ; Int-Vektor lesen
        mov     al,13h          ; Bios Disk-Interrupt
        int     21h             ; ES = Segment von INT 13
        cmp     es:Marker,'LB'  ; installiert?
        je      InsJa           ; ja
        clc                     ; nicht installiert
        jmp     Short InsRet    ;
InsJa:                          ;
        stc                     ; installiert
InsRet:                         ;
        pop     es              ;
        ret                     ;
TestIns EndP

;=============================================================
Hinweis db "                                            ",CR,LF
        db "  BILD 3.0  (C) G. Jürgensmeier  1986-88   ",CR,LF
        db "                                            ",CR,LF
        db "  Bildschirme speichern mit [Alt-Return]   ",CR,LF
        db "  bzw.  mit der im Installationsprogramm   ",CR,LF
        db "  BILDINST angegebenen Tastenkombination   ",CR,LF
        db "                                            ",CR,LF
        db "$"

MelAus  db "BILD: "
        db "Programm ausgeschaltet.",CR,LF,"$"
MelPar  db "BILD: "
        db "Programm bereits installiert.",CR,LF,"$"
MelNein db "BILD: "
        db "Programm nicht installiert, "
        db "kann nicht ausgeschaltet werden.",CR,LF,"$"
MelDos  db "BILD: "
        db "DOS 2.11 oder neuer erforderlich.",CR,LF,"$"

;=============================================================
CodeSeg EndS
        End     Start
```

Listing 1 *(Ende)*

```
masm /b63 /t /ml bild,,nul,nul;
if errorlevel==1 goto exit
link bild;
if errorlevel==1 goto exit
exe2bin bild.exe bild.com
del bild.exe
del bild.obj
:exit
```

Listing 1b: *Stapeldatei zum Assemblieren von* `BILD.ASM`.

```
DEFSNG A-Z
CONST FALSE = 0, TRUE = NOT FALSE
CR$ = CHR$(13): LF$ = CHR$(10)
Esc$ = CHR$(27): NULL$ = CHR$(0)

TYPE BLFile
    Jmp AS STRING * 3
    Marker AS STRING * 2
    Autor AS STRING * 4
    Version AS STRING * 5
    HotKey AS STRING * 1
    HotShift AS STRING * 1
    Ton AS STRING * 1
END TYPE

DIM Bild AS BLFile
DIM Sh$(4), Ky$(88)
RESTORE ShiftMasks
FOR i = 0 TO 4: READ Sh$(i): NEXT i
RESTORE KeyCodes
FOR i = 1 TO 88: READ Ky$(i): PRINT Ky$; : NEXT i


    CLS
    PRINT "                                                    "
    PRINT " BILD Installation 1.00 (C) G.Jürgensmeier 10/88 "
    PRINT "                                                    "

    OPEN "BILD.COM" FOR BINARY ACCESS READ WRITE AS #1
    GET #1, 1, Bild
    IF Bild.Marker <> "BL" THEN GOTO ErrMarker
    IF Bild.Autor <> " GJ " THEN GOTO ErrAutor
    IF Bild.Version <> "10/88" THEN GOTO ErrVersion

    Ton = ASC(Bild.Ton)
    HotKey = ASC(Bild.HotKey)
    HotShift = ASC(Bild.HotShift)
    GOSUB ShowInst

    LOCATE 6, 1, 1
    PRINT "Mit Ton? (J/N/Esc): ";
RepTon:
    a$ = UCASE$(INPUT$(1))
    IF INSTR("JN" + Esc$, a$) = 0 THEN BEEP: GOTO RepTon
    IF a$ = Esc$ THEN GOTO Ende
    PRINT a$: Ton = 0: IF a$ = "J" THEN Ton = 1

    LOCATE 9, 1, 1
    PRINT "Shift-Tasten: "
    FOR i = 0 TO 4: PRINT Sh$(i); "  "; : NEXT: PRINT : PRINT
    PRINT "Tasten:"
    PRINT "A-Z        ÄÖÜ       0-9       <ß'#        +,.-      ";
    FOR i = 1 TO 88
        IF LEN(Ky$(i)) > 1 THEN
            PRINT LEFT$(Ky$(i) + SPACE$(10), 10);
        END IF
    NEXT i
    LOCATE 19, 1
    PRINT "Neue Tastenkombination? ([Return]=keine Änderung) "


RepTaste:
    GOSUB ShowInst
    FOR i = 20 TO 25
        LOCATE i, 1: PRINT SPACE$(79);
    NEXT
    LOCATE 20, 1
    LINE INPUT a$: a$ = UCASE$(RTRIM$(LTRIM$(a$)))
    IF a$ = "" THEN
        HotKey = ASC(Bild.HotKey)
        HotShift = ASC(Bild.HotShift)
    ELSE
        GOSUB Encode
        IF Wrong THEN BEEP: GOTO RepTaste
    END IF
```

Listing 2: *Das Programm* BildInst.

```
    GOSUB ShowInst
    LOCATE 22, 1
    PRINT "Gewählte Taste: "; a$
    PRINT "Installieren mit [Return], abbrechen mit [Esc], ";
    PRINT "[W] = Wiederholen.";
RepAns:
    a$ = UCASE$(INPUT$(1))
    IF a$ <> CR$ AND a$ <> Esc$ AND a$ <> "W" THEN
        BEEP: GOTO RepAns
    END IF
    IF a$ = Esc$ THEN GOTO Ende
    IF a$ = "W" THEN GOTO RepTaste

    Bild.Ton = CHR$(Ton)
    Bild.HotKey = CHR$(HotKey)
    Bild.HotShift = CHR$(HotShift)
    PUT #1, 1, Bild
Ende:
    CLOSE #1
    END



Decode:
    a$ = ""
    IF HotShift = 0 THEN a$ = Sh$(0) + " ": GOTO DecodeEnd
    FOR i = 0 TO 3
        IF (HotShift AND 2 ^ i) THEN
            a$ = a$ + Sh$(i + 1) + " "
        END IF
    NEXT i
DecodeEnd:
    a$ = a$ + Ky$(HotKey)
    RETURN



Encode:
    Wrong = 0: HotKey = 0: HotShift = 0
    a$ = RTRIM$(LTRIM$(UCASE$(a$)))
    WHILE a$ <> ""
        x = INSTR(a$, " "): IF x = 0 THEN x = LEN(a$) + 1
        b$ = LEFT$(a$, x - 1)
        a$ = MID$(a$, x + 1)
        ShFnd = FALSE
        IF b$ = UCASE$(Sh$(0)) THEN
            HotShift = 0: ShFnd = TRUE
            GOTO Trim
        END IF
        FOR i = 1 TO 4
            IF b$ = UCASE$(Sh$(i)) THEN
                HotShift = HotShift OR (2 ^ (i - 1))
                ShFnd = TRUE
            END IF
        NEXT i
        IF ShFnd = TRUE THEN GOTO Trim

        KyFnd = FALSE
        FOR i = 1 TO 88
            IF b$ = UCASE$(Ky$(i)) THEN
                HotKey = i
                KyFnd = TRUE
            END IF
        NEXT i
Trim:
        a$ = LTRIM$(a$)
        IF KyFnd + ShFnd = FALSE THEN
            Wrong = TRUE: GOTO EncodeEnd
        END IF
    WEND
    Wrong = (HotKey = 0)
EncodeEnd:
    IF Wrong THEN
        Ton = ASC(Bild.Ton)
        HotKey = ASC(Bild.HotKey)
        HotShift = ASC(Bild.HotShift)
    END IF
    RETURN
```

Listing 2 *(Fortsetzung)*

```
ShowInst:
    LOCATE 4, 1: PRINT SPACE$(79)
    LOCATE 4, 1
    PRINT "Einstellungen: ";
    PRINT "Ton ";
    IF Ton THEN PRINT "EIN";  ELSE PRINT "AUS";
    PRINT "   Aufruf mit: ";
    GOSUB Decode: PRINT a$
    RETURN


ErrMarker:
    PRINT "Fehler: Keine gültige Kennung in der Programmdatei!"
    CLOSE : END

ErrAutor:
    PRINT "Fehler: Keine gültige Programmdatei!"
    CLOSE : END

ErrVersion:
    PRINT "Fehler: Keine gültige Programmversion!"
    CLOSE : END


ShiftMasks:
    DATA "Normal"    : ' 0
    DATA "RShift"    : ' 1
    DATA "LShift"    : ' 2
    DATA "Ctrl"      : ' 4
    DATA "Alt"       : ' 8


KeyCodes:
    DATA "Esc"       : ' 01
    DATA "1"         : ' 02
    DATA "2"         : ' 03
    DATA "3"         : ' 04
    DATA "4"         : ' 05
    DATA "5"         : ' 06
    DATA "6"         : ' 07
    DATA "7"         : ' 08
    DATA "8"         : ' 09
    DATA "9"         : ' 0A
    DATA "0"         : ' 0B
    DATA "ß"         : ' 0C
    DATA "'"         : ' 0D
    DATA "Backspace" : ' 0E
    DATA "Tab"       : ' 0F
    DATA "Q"         : ' 10
    DATA "W"         : ' 11
    DATA "E"         : ' 12
    DATA "R"         : ' 13
    DATA "T"         : ' 14
    DATA "Z"         : ' 15
    DATA "U"         : ' 16
    DATA "I"         : ' 17
    DATA "O"         : ' 18
    DATA "P"         : ' 19
    DATA "Ü"         : ' 1A
    DATA "+"         : ' 1B
    DATA "Return"    : ' 1C
    DATA ""          : ' 1D Control
    DATA "A"         : ' 1E
    DATA "S"         : ' 1F
    DATA "D"         : ' 20
    DATA "F"         : ' 21
    DATA "G"         : ' 22
    DATA "H"         : ' 23
    DATA "J"         : ' 24
    DATA "K"         : ' 25
    DATA "L"         : ' 26
    DATA "Ö"         : ' 27
    DATA "Ä"         : ' 28
    DATA "<"         : ' 29
    DATA ""          : ' 2A ShiftL
    DATA "#"         : ' 2B
    DATA "Y"         : ' 2C
    DATA "X"         : ' 2D
```

Listing 2 *(Fortsetzung)*

```
DATA "C"         : ' 2E
DATA "V"         : ' 2F
DATA "B"         : ' 30
DATA "N"         : ' 31
DATA "M"         : ' 32
DATA ","         : ' 33
DATA "."         : ' 34
DATA "-"         : ' 35
DATA ""          : ' 36 ShiftR
DATA "PrtSc"     : ' 37
DATA ""          : ' 38 Alternate
DATA "Leertaste": ' 39
DATA "Caps"      : ' 3A
DATA "F1"        : ' 3B
DATA "F2"        : ' 3C
DATA "F3"        : ' 3D
DATA "F4"        : ' 3E
DATA "F5"        : ' 3F
DATA "F6"        : ' 40
DATA "F7"        : ' 41
DATA "F8"        : ' 42
DATA "F9"        : ' 43
DATA "F10"       : ' 44
DATA "NumLock"   : ' 45
DATA "ScrollLock": '46
DATA "Home"      : ' 47
DATA "Oben"      : ' 48
DATA "PgUp"      : ' 49
DATA "/-"        : ' 4A
DATA "Links"     : ' 4B
DATA "/5"        : ' 4C
DATA "Rechts"    : ' 4D
DATA "/+"        : ' 4E
DATA "End"       : ' 4F
DATA "Unten"     : ' 50
DATA "PgDn"      : ' 51
DATA "Ins"       : ' 52
DATA "Del"       : ' 53
DATA "SysReq"    : ' 54
DATA ""          : ' 55
DATA ""          : ' 56
DATA "F11"       : ' 57
DATA "F12"       : ' 58
```

Listing 2 *(Ende)*

```
DEFINT A-Z
CONST False = 0, True = NOT False


    PRINT "ZEIGEBLD 1.00 (C) G. Jürgensmeier 1988"
    PRINT


    dn$ = COMMAND$
    dn$ = UCASE$(LTRIM$(RTRIM$(dn$)))
    IF dn$ = "" THEN PRINT "Kein Dateiname angegeben!": END
    IF INSTR(dn$, ".") = 0 THEN dn$ = dn$ + ".BLD"


    n$ = dn$: GOSUB FileFind
    IF Exist = False THEN
        PRINT "Datei "; dn$; " nicht gefunden!": END
    END IF


    OPEN dn$ FOR BINARY ACCESS READ AS #1
    a$ = SPACE$(4000)
    GET #1, , a$
    CLOSE #1


    CLS
    DEF SEG = &HB000
    FOR i = 0 TO 3999
        x = ASC(MID$(a$, i + 1, 1))
        POKE i, x             ' monochrom
        POKE i + &H8000, x    ' Farbe
    NEXT


    BEEP
    a$ = INPUT$(1)
    LOCATE 24, 1
END


'=================================================================
FileFind:                              ' Datei N$ vorhanden?
    ON ERROR GOTO FileNFound           ' Fehlerroutine
    NAME n$ AS n$                      ' umbenennen
    Exist = True                       ' Datei existiert


FFReturn:
    ON ERROR GOTO 0                    ' Fehlerroutine aus
    RETURN                             ' zurück, EXIST=Status


FileNFound:                            ' Datei nicht vorhanden
    IF ERR = 58 THEN Exist = True: RESUME FFReturn
    Exist = False                      ' Datei existiert nicht
    RESUME FFReturn
```

Listing 3: *Das Programm* ZeigeBld.

Hinweis: *Das in diesem Artikel vorgestellte Programm* Bild *ist eine vereinfachte Version des Programms* Foto *vom Autor, mit dem zusätzlich auch Grafikbildschirme verschiedener Grafikadapter und Textbildschime als Grafiken gespeichert werden können. Mit weiteren Hilfsprogrammen können die Ausgabedateien bearbeitet und in zahlreichen gängigen Grafikformaten gespeichert werden (TIFF, EPS, IMG, Windows-Bitmap, u.a.). Diese Daten können dann in zahlreiche Desktop-Publishing- und Textverarbeitungsprogramme übernommen werden. Dieses Paket ist damit das ideale Werkzeug für jeden, der PC-Programme dokumentieren muß. Eine ausführliche Beschreibung dieses Programmpakets folgt in der nächsten Ausgabe, die darüber hinaus weitere Artikel zum Thema Desktop-Publishing (unter besonderer Berücksichtigung von Programmhandbüchern) enthalten wird.*

Wichtig: *Die Tatsache, daß das hier beschriebene Verfahren in einer Microsoft-Zeitschrift abgedruckt ist, ist keine offizielle Anerkennung derselben. Der Standpunkt von Microsoft bleibt weiterhin: Es wird nicht garantiert, daß diese Möglichkeiten auch in zukünftigen Versionen von DOS existieren.*

```
DEFINT A-Z

    CLS
    PRINT "                                        "
    PRINT "  BLDTXT:                               "
    PRINT "      BLD- in Textdateien umwandeln     "
    PRINT "      (C) Günter Jürgensmeier 10/88     "
    PRINT "                                        "
    PRINT

    LOCATE , , 1

    PRINT "Eingabedatei (BLD): ";
RepIName:
    LINE INPUT a$: a$ = RTRIM$(LTRIM$(a$))
    IF a$ = "" THEN BEEP: GOTO RepIName
    IName$ = a$
    IF (INSTR(IName$, ".") = 0) THEN IName$ = IName$ + ".BLD"

    PRINT "Ausgabedatei (TXT): ";
RepOName:
    LINE INPUT a$: a$ = RTRIM$(LTRIM$(a$))
    IF a$ = "" THEN BEEP: GOTO RepOName
    OName$ = a$
    IF (INSTR(OName$, ".") = 0) THEN OName$ = OName$ + ".TXT"

    PRINT
    PRINT "Datei "; IName$; " wird umgewandelt in ";
    PRINT OName$; ".": PRINT

    ON ERROR GOTO ErrOpen
    OPEN "I", 1, IName$
    CLOSE 1

    OPEN OName$ FOR OUTPUT AS #2
    OPEN IName$ FOR BINARY ACCESS READ AS #1
    Screen$ = SPACE$(4000)
    GET #1, , Screen$
    CLOSE #1

    FOR i = 0 TO 3999 STEP 160
        Lin$ = SPACE$(80): p = 1
        FOR J = 0 TO 159
            IF (i + J) MOD 2 = 0 THEN
                MID$(Lin$, p, 1) = MID$(Screen$, i + J + 1, 1)
                p = p + 1
            END IF
        NEXT
        Lin$ = RTRIM$(Lin$)
EOFCode:
        x = INSTR(Lin$, CHR$(26))
        IF x THEN MID$(Lin$, x, 1) = CHR$(219): GOTO EOFCode

        PRINT Lin$: PRINT #2, Lin$
    NEXT

    CLOSE
    LOCATE 24, 1
END

ErrOpen:
    CLS
    PRINT "BLDTXT: Fehler beim Dateizugriff."
    CLOSE
END
```

Listing 4: *Das Programm* BldTxt.

Datenaustausch zwischen Anwendungsprogrammen über die Zwischenablage:

Datenaustausch unter Windows

Der Datenaustausch zwischen Anwendungsprogrammen stellt seit langem ein Problem für Computer-Benutzer dar. Selbst Neulinge wollen Daten von einem Programm zu einem anderen überspielen. Windows bietet gute Möglichkeiten dazu.

In den frühen Tagen der Personalcomputer war der einzige effektive Weg der manuelle Transfer von Informationen. Die nächste Leistungs- und Funktionalitätsstufe wurde durch die Entwicklung solcher Datei-Standardformate wie Comma Separated Variable (CSV) und Data Interchange Format (DIF) erreicht, die den Datenaustausch mittels Zwischendateien ermöglichten. Später kamen integrierte Pakete auf den Markt, welche die meistgebrauchten Anwendungen in einem einheitlichen Programm zusammenschlossen. Obwohl dies den Bedarf an Datenaustausch zwischen Programmen verringerte, blieb die Grundfrage, wie man Daten austauscht, dennoch bestehen.

Dem Trend zu größerer Leistung und Brauchbarkeit folgend erweiterte Microsoft Windows das Konzept, dessen Pionier der Macintosh war, und bot Entwicklern und Anwendern eine gut zusammenarbeitende Desktop-Umgebung, in der Anwendungen auf natürliche Art kooperieren. Innerhalb von Windows bildeten sich bald zwei Methoden als die De-facto-Mechanismen für Programm-Kommunikation und Datenaustausch heraus: Die Zwischenablage (clipboard) und der dynmische Datenaustausch (Dynamic Data Exchange, DDE).

Von diesen beiden wird die Zwischenablage weit öfters verwendet. Mit der Zwischenablage kann ein Anwender mühelos eine Reihe von Zellen in einer Tabellenkalkulation auswählen, diese in die Zwischenablage kopieren und im Anschluß danach in ein Dokument, das mit einer Textverarbeitung bearbeitet wird, einfügen. Selbst wenn diese zwei Programme von konkurrierenden Softwareherstellern entwickelt wurden, kommunizieren Sie dennoch mühelos miteinander und erscheinen als zusammengehöriges Desktop-System.

Die Zwischenablage ist von Natur aus eine gemeinsame System-Ressource. Zur korrekten Funktion ist es von äußerster Wichtigkeit, daß jede Anwendung mit der Zwischenablage ordnungsgemäß zusammenarbeitet, um so vor Systemabstürzen, endlosen Meldungsschleifen und anderen sonderbaren und verwirrenden Erscheinungen zu schützen. Dieser Artikel wird vom Standpunkt der Entwicklung aus erklären, wie die Zwischenablage richtig verwendet wird und man einige der üblichen Fehler verhindert. Die hier gezeigten Techniken und Einblicke wurden vom Autor während der Erstellung von ClickArt Scrapbook+ der Firma T/Maker Company entwickelt und verwendet, einer leistungsstarken Systemutility, die viele Erweiterungen gegenüber der Windows-Zwischenablage bietet.

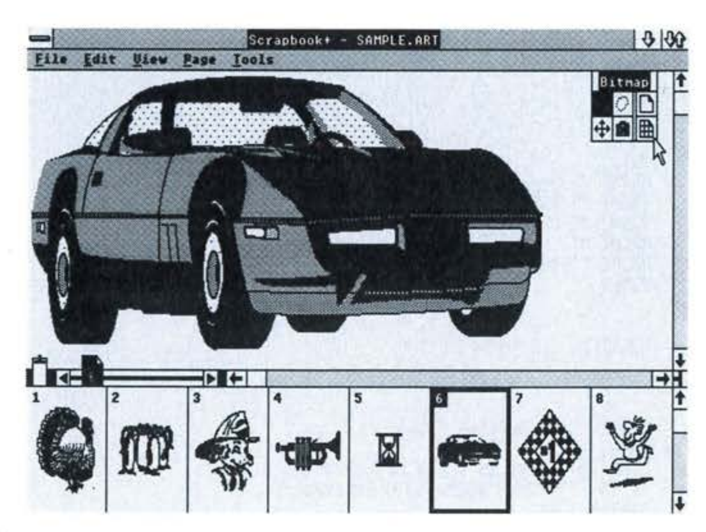


Bild 1: *Eine Anwendung, die die Windows-Zwischenablage nutzt.*

Die Grundlagen der Zwischenablage

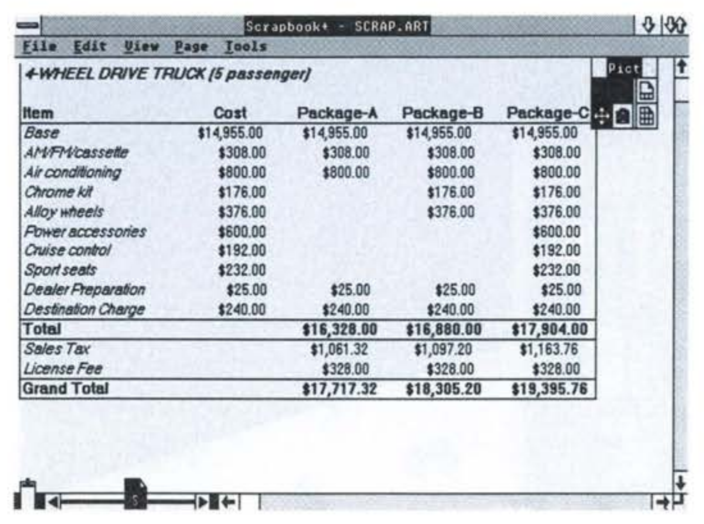
Vom Standpunkt des Anwenders aus ist die Zwischenablage ein temporärer Speicherbereich, auf den von einer Anwendung über das Pull-Down-Menü `Edit` zugegriffen wird. Die folgenden Befehle, wie im Windows Style Guide definiert sind, ermöglichen dem Anwender die Arbeit mit der Zwischenablage:

`Cut/Schneiden`	Kopiere Auswahl in die Zwischenablage und lösche die Auswahl
`Copy/Kopieren`	Kopiere Auswahl in die Zwischenablage
`Paste/Einfügen`	Füge Zwischenablage-Daten in die Anwendungsdaten ein

Unglücklicherweise ist die Sache für den Software-Entwickler etwas komplizierter. Abgesehen von den Cut-, Copy-, und Paste-Operationen, muß der Entwickler mit solchen Problemen kämpfen, wie Zwischenablage-Besitzverhältnis (ownership), Mehrfach-Darstellung (multiple rendering), verzögerte Darstellung (delayed rendering) und die Zwischenablage-Betrachterkette (clipboard viewer chain).

Eine Grundlage der Zwischenablage ist das Zwischenablage-Besitzverhältnis. Weil die Zwischenablage eine gemeinsame System-Ressource ist, läßt Windows sie durch Anwendungen zeitweilig für andere Programme blockieren, solange Veränderungen durchgeführt werden. Nach Abschluß der Veränderungen wird die Zwischenablage sofort freigegeben, sodaß sie für den Zugriff durch andere Anwendungen offen ist.

Vom Konzept her gesehen, scheint die Zwischenablage aus der Sicht des Anwenders immer nur ein Datenelement zur selben Zeit zu enthalten. In Wirklichkeit jedoch ist sie fähig, mehrere Datenelemente gleichzeitig zu bearbeiten, wovon jedes das gleiche Objekt in verschiedenen Formaten darstellen sollte.



4-WHEEL DRIVE TRUCK (5 passenger)

Item	Cost	Package-A	Package-B	Package-C
Base	$14,955.00	$14,955.00	$14,955.00	$14,955.00
AM/FM/cassette	$308.00	$308.00	$308.00	$308.00
Air conditioning	$800.00	$800.00	$800.00	$800.00
Chrome kit	$176.00		$176.00	$176.00
Alloy wheels	$376.00		$376.00	$376.00
Power accessories	$600.00			$600.00
Cruise control	$192.00			$192.00
Sport seats	$232.00			$232.00
Dealer Preparation	$25.00	$25.00	$25.00	$25.00
Destination Charge	$240.00	$240.00	$240.00	$240.00
Total		**$16,328.00**	**$16,880.00**	**$17,904.00**
Sales Tax		$1,061.32	$1,097.20	$1,163.76
License Fee		$328.00	$328.00	$328.00
Grand Total		**$17,717.32**	**$18,305.20**	**$19,395.76**

CF_METAFILEPICT

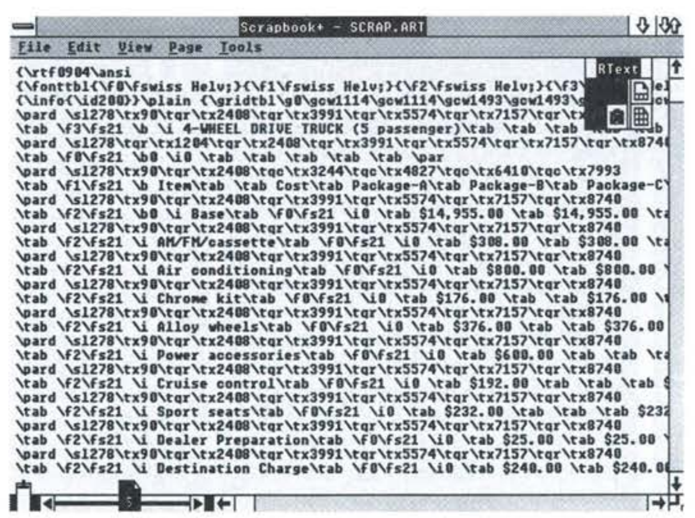
```
TABLE
0,1
"EXCEL"
VECTORS
0,17
""
TUPLES
0,6
""
DATA
0,0
""
-1,0
BOT
1,0
"4-WHEEL DRIVE TRUCK (5 passenger)"
1,0
""
1,0
""
1,0
""
1,0
""
1,0
""
-1,0
BOT
1,0
```

Data Interchange Format (DIF)



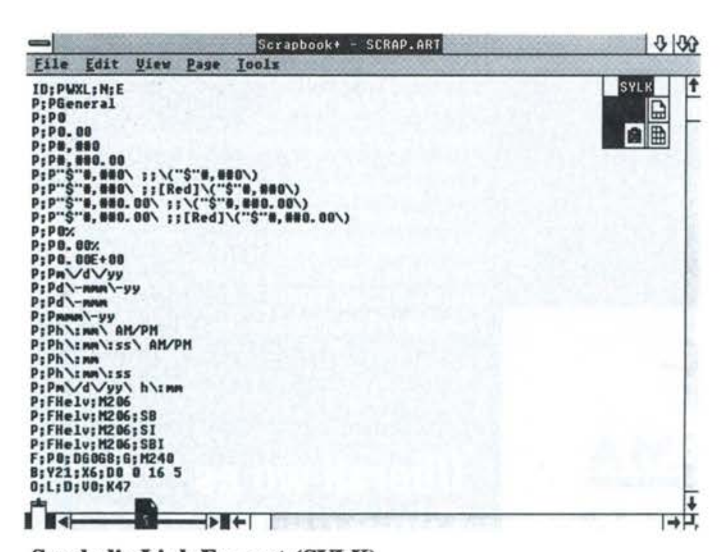
```
4-WHEEL DRIVE TRUCK (5 passenger),,,,,
,,,,,
Item,,Cost,Package-A,Package-B,Package-C
Base,,"$14,955.00 ","$14,955.00 ","$14,955.00 ","$14,955.00 "
AM/FM/cassette,,$308.00 ,$308.00 ,$308.00 ,$308.00
Air conditioning,,$800.00 ,$800.00 ,$800.00 ,$800.00
Chrome kit,,$176.00 ,,$176.00 ,$176.00
Alloy wheels,,$376.00 ,,$376.00 ,$376.00
Power accessories,,$600.00 ,,,$600.00
Cruise control,,$192.00 ,,,$192.00
Sport seats,,$232.00 ,,,$232.00
Dealer Preparation,,$25.00 ,$25.00 ,$25.00 ,$25.00
Destination Charge,,$240.00 ,$240.00 ,$240.00 ,$240.00
Total,,,"$16,328.00 ","$16,880.00 ","$17,904.00 "
Sales Tax,,,"$1,061.32 ","$1,097.20 ","$1,163.76 "
License Fee,,,$328.00 ,$328.00 ,$328.00
Grand Total,,,"$17,717.32 ","$18,305.20 ","$19,395.76 "
```

Comma Separated Variable (CSV)



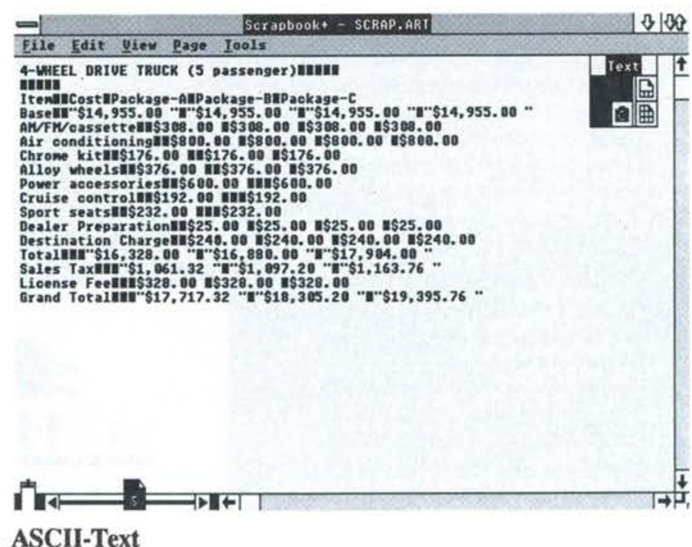
```
{\rtf0904\ansi
{\fonttbl{\f0\fswiss Helv;}{\f1\fswiss Helv;}{\f2\fswiss Helv;}{\f3
{\info{\id200}}\plain {\gridtbl\g0\gcw1114\gcw1114\gcw1493\gcw1493\g
\pard \sl278\tx90\tqr\tx2408\tqr\tx3991\tqr\tx5574\tqr\tx7157\tqr\t
\tab \f3\fs21 \b \i 4-WHEEL DRIVE TRUCK (5 passenger)\tab \tab \tab
\pard \sl278\tqr\tx1204\tqr\tx2408\tqr\tx3991\tqr\tx5574\tqr\tx7157\tqr\tx874
\tab \f0\fs21 \b0 \i0 \tab \tab \tab \tab \tab \par
\pard \sl278\tx90\tqr\tx2408\tqc\tx3244\tqc\tx4827\tqc\tx6410\tqc\tx7993
\tab \f1\fs21 \b Item\tab \tab Cost\tab Package-A\tab Package-B\tab Package-C
\pard \sl278\tx90\tqr\tx2408\tqr\tx3991\tqr\tx5574\tqr\tx7157\tqr\tx8740
\tab \f2\fs21 \b0 \i Base\tab \f0\fs21 \i0 \tab $14,955.00 \tab $14,955.00 \t
\pard \sl278\tx90\tqr\tx2408\tqr\tx3991\tqr\tx5574\tqr\tx7157\tqr\tx8740
\tab \f2\fs21 \i AM/FM/cassette\tab \f0\fs21 \i0 \tab $308.00 \tab $308.00 \t
\pard \sl278\tx90\tqr\tx2408\tqr\tx3991\tqr\tx5574\tqr\tx7157\tqr\tx8740
\tab \f2\fs21 \i Air conditioning\tab \f0\fs21 \i0 \tab $800.00 \tab $800.00
\pard \sl278\tx90\tqr\tx2408\tqr\tx3991\tqr\tx5574\tqr\tx7157\tqr\tx8740
\tab \f2\fs21 \i Chrome kit\tab \f0\fs21 \i0 \tab $176.00 \tab \tab $176.00 \
\pard \sl278\tx90\tqr\tx2408\tqr\tx3991\tqr\tx5574\tqr\tx7157\tqr\tx8740
\tab \f2\fs21 \i Alloy wheels\tab \f0\fs21 \i0 \tab $376.00 \tab \tab $376.00
\pard \sl278\tx90\tqr\tx2408\tqr\tx3991\tqr\tx5574\tqr\tx7157\tqr\tx8740
\tab \f2\fs21 \i Power accessories\tab \f0\fs21 \i0 \tab $600.00 \tab \tab \t
\pard \sl278\tx90\tqr\tx2408\tqr\tx3991\tqr\tx5574\tqr\tx7157\tqr\tx8740
\tab \f2\fs21 \i Cruise control\tab \f0\fs21 \i0 \tab $192.00 \tab \tab \tab $
\pard \sl278\tx90\tqr\tx2408\tqr\tx3991\tqr\tx5574\tqr\tx7157\tqr\tx8740
\tab \f2\fs21 \i Sport seats\tab \f0\fs21 \i0 \tab $232.00 \tab \tab \tab $232
\pard \sl278\tx90\tqr\tx2408\tqr\tx3991\tqr\tx5574\tqr\tx7157\tqr\tx8740
\tab \f2\fs21 \i Dealer Preparation\tab \f0\fs21 \i0 \tab $25.00 \tab $25.00
\pard \sl278\tx90\tqr\tx2408\tqr\tx3991\tqr\tx5574\tqr\tx7157\tqr\tx8740
\tab \f2\fs21 \i Destination Charge\tab \f0\fs21 \i0 \tab $240.00 \tab $240.0
```

Rich Text Format (RTF)



```
ID;PWXL;N;E
P;PGeneral
P;P0
P;P0.00
P;P#,##0
P;P#,##0.00
P;P"$"#,##0\ ;;\("$"#,##0\)
P;P"$"#,##0\ ;;[Red]\("$"#,##0\)
P;P"$"#,##0.00\ ;;\("$"#,##0.00\)
P;P"$"#,##0.00\ ;;[Red]\("$"#,##0.00\)
P;P0%
P;P0.00%
P;P0.00E+00
P;Pm\/d\/yy
P;Pd\-mmm\-yy
P;Pd\-mmm
P;Pmmm\-yy
P;Ph\:mm\ AM/PM
P;Ph\:mm\:ss\ AM/PM
P;Ph\:mm
P;Ph\:mm\:ss
P;Pm\/d\/yy\ h\:mm
P;FHelv;M206
P;FHelv;M206;SB
P;FHelv;M206;SI
P;FHelv;M206;SBI
F;P0;DG0G8;G;M240
B;Y21;X6;D0 0 16 5
O;L;D;V0;K47
```

Symbolic Link Format (SYLK)



```
4-WHEEL DRIVE TRUCK (5 passenger)■■■■■
■■■■■
Item■■Cost■Package-A■Package-B■Package-C
Base■■"$14,955.00 "■"$14,955.00 "■"$14,955.00 "■"$14,955.00 "
AM/FM/cassette■■$308.00 ■$308.00 ■$308.00 ■$308.00
Air conditioning■■$800.00 ■$800.00 ■$800.00 ■$800.00
Chrome kit■■$176.00 ■■$176.00 ■$176.00
Alloy wheels■■$376.00 ■■$376.00 ■$376.00
Power accessories■■$600.00 ■■■$600.00
Cruise control■■$192.00 ■■■$192.00
Sport seats■■$232.00 ■■■$232.00
Dealer Preparation■■$25.00 ■$25.00 ■$25.00 ■$25.00
Destination Charge■■$240.00 ■$240.00 ■$240.00 ■$240.00
Total■■■"$16,328.00 "■"$16,880.00 "■"$17,904.00 "
Sales Tax■■■"$1,061.32 "■"$1,097.20 "■"$1,163.76 "
License Fee■■■$328.00 ■$328.00 ■$328.00
Grand Total■■■"$17,717.32 "■"$18,305.20 "■"$19,395.76 "
```

ASCII-Text

Bild 2: *Mehrfach dargestellte Zwischenablage-Daten.*

Dieses Konzept der Mehrfachdarstellung ermöglicht es Anwendungen wie etwa Microsoft Excel, die Zwischenablage bei einer Kopieroperation mit bis zu 13 verschiedenen Formaten oder Darstellungen der gleichen Information zu beliefern. Die einfügende Anwendung kann dann das gewünschte Format aus den in der Zwischenablage vorhandenen auswählen. *Bild 2* zeigt ein Beispiel für mehrfach dargestellte Daten.

Wenn die Zwischenablage mehrfach dargestellte Daten enthält, bleibt es der Entscheidung der einfügenden Anwendung überlassen, welches Format am besten den Bedürfnissen des Benutzers entspricht. Raffinierte Programme plazieren Daten meistens in einer nach Priorität geordneten Liste in der Zwischenablage, beginnend mit dem Format höchster Informationsdichte. Gewöhnlich jedoch ist die Reihenfolge zufällig, und man sollte sich nicht darauf verlassen. Die einfügende Anwendung kann dann diese Formate aufreihen und eine alternative Darstellung erhalten, nach Priorität geordnet, wie vom Urheber gewünscht.

Man kann sich leicht vorstellen, daß das Arbeiten mit mehrfach dargestellten Zwischenablage-Daten ungeheuer viel kostbaren Speicherplatz verschlingen kann. Glücklicherweise versucht ein anderes Konzept, als verzögerte Darstellung bezeichnet, die Speicherplatzzuweisung auf jene Formate zu reduzieren, die auch wirklich gebraucht werden. Die verzögerte Darstellung erlaubt der kopierenden Anwendung eine »Zusage« für jede mögliche Darstellung in die Zwischenablage zu setzen. Diese Zusagen verbleiben in der Zwischenablage bis eine Anwendung ein bestimmtes Datenelement anfordert. Wenn diese Anfoderderung empfangen wird, muß die kopierende Anwendung die Zusage durch Lieferung der Daten im angeforderten Format erfüllen, auch wenn der Kopiervorgang dieser Daten in die Zwischenablage schon länger her ist. Das Endergebnis ist, daß Datenbereiche nur für solche Formate zugewiesen werden, die man auch wirklich braucht, und dies führt dazu, daß weniger Systemspeicherplatz benötigt und die Berechnungen eingespart werden, die man zur Erstellung unerwünschter Formate brauchen würde. Die verzögerte Darstellung erspart überdies Verarbeitungszeit für einfach dargestellte Elemente.

Zu den mehrfachen und verzögerten Darstellungskonzepten bietet die Windows-Zwischenablage einen Mechanismus, wodurch beteiligte Anwendungen automatisch benachrichtigt werden können, wenn Veränderungen in der Zwischenablage auftreten. Diese Einrichtung, die Zwischenablage-»Betrachterkette« genannt und in *Bild 3* gezeigt wird, veranlaßt das System eine `WM_DRAWCLIPBOARD`-Meldung an alle beteiligten Fenster zu senden. Der Empfang einer solchen Meldung weist darauf hin, daß eine Veränderung der Zwischenablage aufgetreten ist, die von Interesse für die Anwendung sein kann. Nach Verarbeitung der Meldung ist die Verantwortung zur Weitergabe der Meldung an die nächste Anwendung in der Kette ebenfalls der Anwendung überlassen.

Funktion	Beschreibung
ChangeClipboardChain	Entfernt ein Fenster aus der Zwischenablage-Betrachterkette.
CloseClipboard	Schließt die Zwischenablage.
CountClipboardFormats	Gibt die Anzahl der verfügbaren Formate in der Zwischenablage zurück. Die Zwischenablage braucht nicht offen sein.
EmptyClipboard	Leert die Zwischenablage und ordnet das Zwischenablage-Besitzrecht neu zu. Die Zwischenablage muß offen sein.
EnumClipboardFormats	Zählt die verfügbaren Zwischanbalgae-Formate durch. Die Zwischenablage muß offen sein.
GetClipboardData	Liest die Daten aus der Zwischenabalage (im angefragten Format). Die Zwischenablage muß offen sein.
GetClipboardFormatName	Stellt den ASCII-Namen eines nicht vordefinierten Formats fest.
GetClipboardOwner	Stellt die Fenster-Handle fest, die mit dem aktuellen Zwischenablage-Besitzer verbunden ist.
GetClipboardViewer	Stellt die Handle des ersten Fensters in der Zwischenablage-Betrachterkette fest.
IsClipboardFormatAvailable	Liefert TRUE, wenn die Daten im angegebenen Format in der Zwischenabalage verfügbar ist.
OpenClipboard	Öffnet die Zwischenablage.
RegisterClipboardFormat	Registriert ein neues Zwischenablage-Format.
SetClipboardData	Kopiert eine Daten-Handle in die Zwischenablage. Die Zwischenablage muß offen sein.
SetClipboardViewer	Fügt eine Fenster-Handle in die Zwischenablage-Betrachterkette an.

Tabelle 1: *Die verfügbaren Funktionen für die Zwischenablage.*

Die Programmierschnittstelle der Zwischenablage

Anwendungen, die die Zwischenablage verwenden wollen, können dies durch Aufruf von Funktionen erreichen, die vom Windows Application Programming Interface (API) (= Anwendungs-Programmier-Schnittstelle) geliefert werden und durch Bearbeitung bestimmter dazugehöriger Meldungen. Die *Tabelle 1*, die zum Teil dem Windows Software Development Kit entnommen wurde, beschreibt kurz die verfügbaren Funktionen. Außer diesen Funktionen werden auch die Zwischenablage-bezogenen Meldungen, aufgelistet in *Tabelle 2*, von Windows unterstützt.

Datenformate in der Zwischenablage

Windows besitzt sieben vordefinierte Formate, die von Standard-ASCII-Text bis Grafikdaten in Binärformat reichen. Außer diesen vordefinierten Formaten bietet Windows einen Mechanismus, mit dem man zusätzliche benutzerdefinierte Formate erstellen kann. Unter diesen benutzerdefinierten Erweiterungen sind sechs, die von den meisten bedeutenden Anwendungen unterstützt werden. *Tabelle 3* beschreibt kurz jedes dieser Formate. Zu beachten ist, daß sich alle Formate, deren Namen mit der Präfix `CF_` anfangen, auf vordefinierte Zwischenablage-Formate beziehen.

Meldung	Beschreibung
WM_ASKCBFORMAT-NAME*	Fragt nach dem Namen des CF_OWNERDISPLAY Formats.
WM_CHANGE-CBCHAIN	Teilt Mitgliedern der Betrachterkette eine Veränderung in der Kette mit.
WM_DESTROY-CLIPBOARD	Signalisiert dem Zwischenablage-Besitzer, daß der Zwischenablage-Inhalt zerstört wird.
WM_DRAWCLIP-BOARD	Teilt einer Anwendung in der Betrachterkette eine Veränderung in der Zwischenablage mit.
WM_HSCROLL-CLIPBOARD*	Fordert horizontalen Bildlauf für das Format CF_OWNERDISPLAY an.
WM_PAINT-CLIPBOARD*	Fordert die Anzeige des Formats CF_OWNERDISPLAY an.
WM_RENDER-ALLFORMATS	Teilt dem Zwischenablage-Besitzer mit, daß er alle zugesagten Zwischenablage-Daten darstellen muß.
WM_RENDER-FORMAT	Teilt dem Zwischenablage-Besitzer mit, daß er die in die Zwischenablage kopierten Daten formatieren muß.
WM_SIZE-CLIPBOARD*	Teilt dem Zwischenablage-Besitzer mit, daß sich die Größe des Betrachter-Fensters verändert hat.
WM_VSCROLL-CLIPBOARD*	Fordert vertikalen Bildlauf für das Format CF_OWNERDISPLAY an.
*	Bezieht sich auf Nachrichten, die mit dem Format CF_OWNERDISPLAY zu tun haben.

Tabelle 2: *Zwischenablage-bezogene Meldungen.*

Benutzerdefinierte Zwischenablage-Formate werden durch Windows registriert, indem man `RegisterClipboardFormat` mit einer eindeutigen Zeichenkette aufruft. Jede Anwendung, die dieses Format verwenden will, muß es neu registrieren und erhält dadurch die selbe Format-Handle. Wenn einmal definiert, können andere Anwendungen diesen registrierten Namen durch Aufruf der Funktion `GetClipboardFormatName` mit der Format-Handle als Parameter abfragen. Zu beachten ist, daß der Aufruf von `GetClipboardFormatName` mit einem vordefinierten Zwischenablage-Format einen NULL-String zurückliefert. Des weiteren, daß bei der Verwendung eines `CF_OWNERDISPLAY`-Objektes der Name des Formats durch Senden der Meldung `WM_ASKCBFORMATNAME` an den Zwischenablage-Besitzer zurückgeholt werden kann.

Format	Beschreibung
CF_BITMAP*	Windows GDI-Bitmap
CF_DIF*	Data Interchange Format von Software Arts
CF_METAFILE-PICT*	Windows Metadatei-Bilder (METAFILEPICT)
CF_OWNER-DISPLAY	Benutzereigenes Bildformat
CF_SYLK*	Microsoft Symbolic Link Format (Kalkulationsdaten)
CF_TEXT*	Normaler ASCII-Text
CF_TIFF*	Tagged Image File Format
BIFF	Binary Interchange Data Format
CSV*	Durch Semikolon getrennte Variablen (comma separated variables)
Postscript*	PostScript-Text
Printer_Bitmap*	Drucker-Bitmap (erstmals bei Microsoft Excel)
Printer_Picture*	Drucker Metadatei-Bilder (erstmals von Excel definiert)
Rich Text*	Rich Text Format
[priv. Formate]	Andere benutzerdefinierte Formate
*	Weist auf Formate hin, die von Scrapbook+ verarbeitet werden.

Tabelle 3: *Datenformate der Zwischenablage.*

Von den gewöhnlich verwendeten Zwischenablage-Formaten werden bis auf `CF_BITMAP` und `Printer_Bitmap` alle durch Handles auf globale Datenblöcke des Systemspeicherplatzes repräsentiert. Speicherplatz für diese Formate wird durch den Gebrauch von `GlobalAlloc` mit dem Flag `GMEM_MOVEABLE` als Parameter zugewiesen. Wenn die sich ergebende Format-Handle an die Zwischenablage übergeben wird (mit einem Aufruf von `SetClipboardData`), ändert das System die Speicherattribute mit `GlobalReAlloc` nach `GMEM_DDESHARE` und übergibt das Besitzrecht des Blocks an Windows. Dadurch bleibt der Speicherbereich erhalten, nachdem die kopierende Anwendung verlassen wurde, obwohl unter normalen Umständen der Speicherplatz einer Anwendung nach dem Verlassen zum Entfernen freigegeben wird. Zu beachten ist, daß nach der Übertragung eines Objekts durch eine Anwendung in die Zwischenablage das Objekt in den Besitz des Systems übergeht und die Anwendung die Daten nicht mehr weiterbehandeln sollte.

Nun werfen wir einen Blick auf Zwischenablage-Formate und analysieren deren potentiellen Gebrauch und Mißbrauch.

CF_BITMAP. Dieses Format wird zum Übertragen von GDI-BitMap-Bildern verwendet. Bitmaps können mit `CreateBitmap`, `CreateBitmapIndirect` oder `CreateCompatibleBitmap` erzeugt werden. Die resultierende Format-Handle kann mit `SetClipboardData` ohne weitere Änderungen in die Zwischenablage gestellt werden. Zu beachten ist, daß die Funktion `CreateDiscardableBitmap` bei der Erzeugung von Bitmaps für den Transfer über die Zwischenablage nicht verwendet werden darf.

Wegen des GDI gibt es das CF_BITMAP-Format in zwei Versionen. Die erste ist eine konventionelle monochrome Bitmap. Die Anordnung der Bytes ist fest und unterstützt die direkte Bearbeitung der Bitmap-Daten. Außerdem erlaubt diese standardisierte Byte-Anordnung die Übergabe von Monochrom-Bitmaps von einer Maschine zur anderen, jedoch in Abhängigkeit von den Darstellungs- und Aspektverhältnissen. Die zweite Version ist die Farb-Bitmap, die leider in Windows strikt geräteabhängig ist und die Portabilität eines solchen Objektes ist nicht sichergestellt. Darüber hinaus müssen Farb-Bitmaps als Objekte mit unbekannter Byte-Anordnung angesehen werden, was die direkte Manipulation der Daten ausschließt.

CF_DIF. Das DIF-Format von Software Arts besteht aus ASCII-Text, dessen Zeilen durch das Zeichenpaar Carriage Return/Line Feed (CR/LF) beendet sind. Im Gegensatz zu `CF_TEXT` ist der DIF-Datenblock nicht mit Null abgeschlossen, da das Ende in den Daten codiert ist. Der einfachste Weg zur Bestimmung der Größe der Daten ohne Analyse der Informationen ist der Gebrauch von `GlobalSize` zur Abfrage der Blockgröße. Mit dieser Methode kann eine einzige Funktion geschrieben werden, die mehrere verschiedene textbezogene Zwischenablage-Formate verarbeitet.

Impressum

Das *Microsoft System Journal* erscheint alle zwei Monate (ungerade Monatszahlen) etwa Mitte des Vormonats.

Herausgeber, verantwortlich und Anschrift der Redaktion:
Microsoft GmbH, Redaktion *Microsoft System Journal*,
Erdinger Landstr. 2, D-8011 Aschheim-Dornach
Telefon: 089 / 46107-0, Teletex/Telex: (17) 89 83 28, Telefax: 90 63 55

Redaktion:
Günter Jürgensmeier, Haar und Hartmut Niemeier, Wildenberg
Mitarbeiter dieser Ausgabe:
Michael Bülow, Kaare Christian, Günter Jürgensmeier, Hartmut Niemeier, Matt Trask, Kevin P. Welch, David E. West, Richard Wilton

Manuskripteinsendungen:
Manuskripte und Programmlistings werden von der Redaktion gerne angenommen. Mit der Einsendung von Manuskripten und Listings gibt der Verfasser die Zustimmung zum Abdruck und zur Vervielfältigung der Programmlistings auf Datenträgern. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte und Listings wird keine Haftung übernommen. Nicht zur Veröffentlichung gelangte Manuskripte und Listings können nur zurückgeschickt werden, wenn Rückporto beiliegt.

Titelgestaltung: Hermann Menig
Anzeigenverkauf: Marianne Nuß
Druck und Abonnements:
schury praxisformulare GmbH, Postfach 270, D-8200 Rosenheim

Bezugspreise: Das Einzelheft kostet DM 19,80. Der Abonnementpreis beträgt DM 115,- für 6 Ausgaben und DM 210,- für 12 Ausgaben. Zu den einzelnen Ausgaben ist zum Preis von DM 19,80 eine Diskette mit allen Listings erhältlich. Das Abonnement inklusive Diskette kostet DM 230,- für 6 Ausgaben und DM 420,- für 12 Ausgaben. In den Preisen enthalten sind Mehrwertsteuer, Versandkosten und Zustellgebühren. Auslandsbezug auf Anfrage. Sollte die Zeitschrift aus Gründen, die nicht vom Herausgeber zu vertreten sind, nicht geliefert werden können, besteht kein Anspruch auf Nachlieferung oder Erstattung vorausbezahlter Bezugsgelder.
Bezugsmöglichkeiten: In Buchhandlungen und im Computer-Fachhandel. Abonnements und Einzelbestellungen: schury GmbH.

Für die Programme, die als Beispiele veröffentlicht werden, kann der Herausgeber weder Gewähr noch irgendwelche Haftung übernehmen. Aus der Veröffentlichung kann nicht geschlossen werden, daß die beschriebenen Lösungen oder verwendeten Bezeichnungen frei von gewerblichen Schutzrechten sind. Die Erwähnung oder Beurteilung von Produkten stellt, soweit es sich nicht um Microsoft-Produkte handelt, keine irgendwie geartete Empfehlung der Microsoft GmbH dar. Für die mit Namen oder Signatur gekennzeichneten Beiträge übernimmt der Herausgeber lediglich die presserechtliche Verantwortung.

Das *Microsoft System Journal* wird mit Microsoft Word 4.0 geschrieben und gestaltet. Der Ausdruck erfolgt mit den HP-Softfonts AD und AF und dem Programm- und Schriftenpaket *DocuJet* auf einem HP-LaserJet Series II.

```
/* Lokale Daten */

LPSTR              lpMemory;
LPMETAFILEPICT     lpMetaFile;

/* Definiert Anzahl der Formate */
wFormats = 3;

/* Definiert Textdaten */
Clipboard[0].wFormat = CF_TEXT;
Clipboard[0].hGlobalData = GlobalAlloc( GHND, (DWORD)100 );
if ( Clipboard[0].hGlobalData ) {
   lpMemory = GlobalLock( Clipboard[0].hGlobalData );
   lstrcpy( lpMemory, "Some sample text for the clipboard." );
   GlobalUnlock( Clipboard[0].hGlobalData );
} else
<...den Benutzer warnen - zu wenig Speicher...>

/* Definiert Bitmap-Daten */
Clipboard[1].wFormat = CF_BITMAP;
Clipboard[1].hGlobalData = CreateBitmap( 50, 50, 1, 1, NULL);
if ( Clipboard[1].hGlobalData ) {
<...Bitmap-Bits definieren...>
} else
<...den Benutzer warnen - zu wenig Speicher...>

/* Definiert Metadatei-Daten */
Clipboard[2].wFormat = CF_METAFILEPICT;
Clipboard[2].hGlobalData = GlobalAlloc( GHND,
(DWORD)sizeof(METAFILEPICT) );
if ( Clipboard[2].hGlobalData ) {
   lpMetaFile = (LPMETAFILEPICT)GlobalLock(
                    Clipboard[2].hGlobalData );
   lpMetaFile->mm = MM_ANISOTROPIC;
   lpMetaFile->xExt = 0;
   lpMetaFile->yExt = 0;
   lpMetaFile->hMF = <...Metadatei definieren...>
                    <... und initialisieren ...>
   GlobalUnlock( Clipboard[2].hGlobalData );
} else
<...den Benutzer warnen - zu wenig Speicher...>
```

Listing 1: *Definition globaler Datenblöcke.*

```
/* Definiert Anzahl der Formate */
wFormats = 3;

/* Definiert Textdaten */
Clipboard[0].wFormat = CF_TEXT;
Clipboard[0].hGlobalData = NULL;

/* Definiert Bitmap-Daten */
Clipboard[1].wFormat = CF_BITMAP;
Clipboard[1].hGlobalData = NULL;

/* Definiert Metadatei-Daten */
Clipboard[2].wFormat = CF_METAFILEPICT;
Clipboard[2].hGlobalData = NULL;
```

Listing 2: *Datenformat-Definitionen*

CF_METAFILEPICT. Dieses Format bezieht sich auf einen globalen Speicherbereich, der eine Struktur vom Typ `METAFILEPICT` enthält. Diese Struktur enthält vier Felder: mm, der Mapping-Modus, der bei der Anzeige der Metadatei gebraucht wird. `xExt` und `yExt`, von denen die gewünschte Größe des Bildes, das von der Metadatei gezeichnet wird, berechnet werden kann (siehe im Windows SDK Programmer's Reference Manual für weitere Informationen); und `hMF`, eine Daten-Handle für die wirkliche Metadatei. Um eine Kopie dieser Daten zu machen, verwendet man `GlobalAlloc` zur Speicherzuweisung für die `METAFILEPICT`-Struktur, danach `CopyMetafile`, um eine Kopie der Metadatei anzufertigen, auf die sich das `hMF`-Feld bezieht.

CF_OWNERDISPLAY. Das Zwischenablage-Format `CF_OWNERDISPLAY` ermöglicht Anwendungen ihre eigene Bilddarstellung interner Daten beizubehalten. Damit solche Informationen angezeigt werden können, ist der Zwischenablage-Betrachter darauf beschränkt, Meldungen an die jeweilige Anwendung zu senden, wenn das Fenster neu angezeigt werden soll.

Diese ungewöhnliche Darstellungsmethode verwendet die Anwendung Zwischenablage (ABLAGE.EXE) dazu, Write-formatierten Text darzustellen. Beim Gebrauch dieses Formats erhält der Zwischenablage-Besitzer eine Serie von Meldungen, die normalerweise mit denen übereinstimmt, auf die der Zwischenablage-Betrachter stößt. Der Besitzer ist daher für die nötige Durchführung der zuständigen Aktion verantwortlich, wann immer eine Zeichen-, Größen-, oder Bildlauf-Operation auftritt.

Eine ärgerliche Eigenschaft des `CF_OWNERDISPLAY`-Formats ist die Art, auf die der Arbeitsbereich des Zwischenablage-Betrachters gehandhabt wird. Im Falle von Windows Write wird automatisch der Stil des Fensters geändert, um einen zusätzlichen Satz von horizontalen und vertikalen Bildlaufleisten einzufügen, unabhängig von der Tatsache, daß diese schon als Unterfenster vorhanden sein könnten. Darüberhinaus sind die `CF_OWNERDISPLAY`-Daten, die von den meisten Anwendungen geliefert werden, besondere interne Formate und von wenig Nutzen für andere Anwendungen.

CF_SYLK. Dieses Zwischenablage-Format, eine Speicherdarstellung des SYLK-Dateiformats, gleicht dem Format `CF_DIF`, in welchem jede Zeile eines ASCII-Textes durch ein CR/LF-Paar abgeschlossen wird. Ebenso wie bei `CF_DIF` ist der Datenblock nicht mit Null abgeschlossen. Daher sollte `GlobalSize` zum Feststellen der Blockgröße verwendet werden, unter Berücksichtigung der üblichen 16-Byte-Paragraph-Einschränkungen.

CF_TEXT. Das Format `CF_TEXT` ist eines der einfachsten der Zwischenablage. Jede Textzeile wird mit einem CR/LF-Paar abgeschlossen und der gesamte Textblock mit einem Null-Zeichen. Jede Anwendung, die den Datentransfer von Daten im Textformat über die Zwischenablage verwendet, sollte den Gebrauch dieses Formats zusätzlich zu jedem anderen, wie Rich Text, das zum Datenaustausch von Zeichen-, Paragraph-, und Dokument-Formatierinformationen dient, erwägen.

```
/* Lokale Variablen */
WORD   i;
if ( OpenClipboard(hWnd) ) {
   EmptyClipboard();
   for ( i=0; i<wFormats; i++ )
      SetClipboardData( Clipboard[i].wFormat,
                        Clipboard[i].hGlobalData );
} else
<...den Benutzer warnen - Zwischenablage kann nicht geöffnet
werden...>
```

Listing 3: *Programmausschnitt für die Zwischenablage-Kopieroperation.*

CF_TIFF. Das Format `CF_TIFF` wird zum Transfer von Monochrom-, Graustufen-, oder Farbbildern zwischen zusammenarbeitenden Anwendungen verwendet. Obgleich ursprünglich für den Einsatz bei sehr großen Bildern entwickelt, wird dieses Formats nun immer mehr für Bilder jeglicher Größe eingesetzt, da es ein exaktes Speicherabbild der originalen Microsoft/Aldus-Dateispezifikationen darstellt. Aufgrund der Speicherbeschränkungen sollten Anwendungen, die dieses Format unterstützen, auch fähig sein, die TIFF-Daten direkt von Platten zu lesen.

In den letzten Monaten haben mehrere Software-Entwickler Erweiterungen für die Spezifikation des Zwischenablage-Formats `CF_TIFF` vorgeschlagen, die eine Definition eines dateiunterstützten Mechanismus für den Zwischenablage-Transfer von großen Bildern beinhalten soll. Dieser Plan wird wohl kaum von Windows unterstützt werden, da dies Modifikationen der internen Zwischenablage-Verwaltung zur Bearbeitung der dazugehörigen, auf Anwendungen aufgeteilten temporären Dateien, erfordern würde. Weitere Informationen über die `CF_TIFF`-Datei und die Zwischenablage-Spezifikation erhält man im Windows SDK Extensions Manual.

BIFF. Dieses benutzerdefinierte Format wird derzeit von mehreren Anwendungen unterstützt, wovon die bedeutendste Microsoft Excel ist. BIFF (für Binary File Format) ist das Dateiformat in welchem Excel-Dokumente auf der Platte abgelegt werden. Eine BIFF-Datei ist ein vollständiges in sich geschlossenes Paket, das aus einer Reihe von Datensätzen variabler Länge besteht. Zum Beispiel beschreibt ein Datensatz die Größe und Lage eines Abbildungs-Fensters in einem Dokument, ein anderer die Formel für eine Zelle und ein dritter das Format eines Bildes.

Obwohl die einzelnen BIFF-Datensatz-Typen verschiedene Informationen enthalten, folgt doch jeder Datensatz dem selben Format: Typ des Datensatzes, Länge des Datensatzes, Datensatzinhalte (variable Länge).

Das BIFF-Zwischenablage- und Dateiformat wird in der Zukunft ein Standard werden, wie es auch TIFF wurde. Einige unabhängige Software-Entwickler unterstützen schon dieses Format für den Transfer von binären Finanzdaten zwischen Anwendungen. Weitere Informationen über dieses Datei- und Zwischenablage-Format sind über Microsoft zu bekommen.

CSV. Software-Entwickler kennen schon seit langem das CSV-Dateiformat. Es ist vielleicht der kleinste gemeinsame Nenner für Datenaustauschformate (andere meinen, das sei das Textformat). Von Microsoft Excel für die Zwischenablage eingeführt, wird dieses benutzerdefinierte Format in Windows genauso wie `CF_TEXT` behandelt. Die ASCII-Textzeilen sind durch CR/LF-Paare getrennt und der gesamte Block ist durch ein Null-Zeichen abgeschlossen.

In jeder Textzeile sind alle Datenelemente durch Kommas getrennt und Text-Zeichenketten in Anführungszeichen eingeschlossen. Wegen seiner Einfachheit wird CSV auf breiter Basis verwendet und jede Anwendung, die Daten verarbeitet, die in diesem Format sinnvoll sind, sollte diesen Standard unterstützen.

PostScript. Encapsulated PostScript (EPS) ist ein sich herausbildender Standard für den Austausch von Texten und Daten zwischen zusammenarbeitenden Anwendungen. Das benutzerdefinierte PostScript-Format wurde von seinen Autoren zum Transfer von EPS-Bildern über die Zwischenablage entwickelt. Zur Zeit wird es von Scrapbook+ und einigen anderen Windows-Anwendungen unterstützt. Der PostScript-Text ist gewöhnlich von einer oder mehreren Darstellungen begleitet, typischerweise einem Bild im Format `CF_TIFF` oder `CF_METAFILEPICT`.

Das PostScript-Format ist im Konzept identisch mit `CF_TEXT`, außer daß jede Textzeile mit einem alleinigen CR abgeschlossen ist und der gesamte Block mit einem Null-Zeichen. Mehr Informationen über den Transfer von EPS-Bildern über die Zwischenablage sind im folgenden Artikel »Encapsulated PostScript« zu finden.

Printer_Bitmap. Dieses benutzerdefinierte Format ist identisch mit `CF_BITMAP`, außer daß es durch Ausgabemöglichkeiten des gewählten Druckers beschränkt ist. Zum Beispiel würde eine in diesem Format bearbeitete Farb-Bitmap monochrom wiedergegeben werden, wenn als Ausgabe-Drucker ein Laser-Drucker definiert wäre. Der Gebrauch dieses Formats ermöglicht der kopierenden Funktion alle wichtigen Umsetzungen durchzuführen, die für andere Programme schwierig sind.

Printer_Picture. Das Format `Printer_Picture` ist wie `Printer_Bitmap` identisch mit `CF_METAFILEPICT`, aber beschränkt auf die Ausgabemöglichkeiten des gewählten Druckers. Die Anwendung, die dieses Format übergibt, ist verantwortlich für die Durchführung aller nötigen Umsetzungen für das vorgegebene Ausgabegerät. Obgleich sich das in der Theorie einfach anhört, kann die Definition dieses Formats schwierig sein, wenn es das Übertragen von Schriften, Farben, und anderen GDI-Objekten zu einem spezifischen Ausgabegerät erfordert.

```
LONG FAR PASCAL WndProc(
      HWND         hWnd,
      WORD         wMessage,
      WORD         wParam,
      LONG         lParam
   )
{
   /* Lokale Variablen */
   WORD             i;
   LPSTR            lpMemory;
   LPMETAFILEPICT   lpMetaFile;
   /* Meldungsverarbeitung */
   switch( wMessage )
      {
   case WM_RENDERFORMAT :
      /* Gibt ein einzelnes Datenelement wieder */
      /* Sucht nach Eintrag in der Tabelle
                  wParam = erfordertes Format */
      for ( i=0; (i<wFormats) && (Clipboard[i].wFormat !=
                                  wParam); i++ );
      if ( i < wFormats ) {
         /* Definiert Daten */
         switch( wParam )
            {
         case CF_TEXT : /* Fragt nach Textdaten */
            Clipboard[i].hGlobalData =
               GlobalAlloc( GHND, (DWORD)100 );
            if ( Clipboard[i].hGlobalData ) {
               lpMemory =GlobalLock(Clipboard[i].hGlobalData );
               lstrcpy( lpMemory, "Some sample text for the
                              clipboard." );
               GlobalUnlock( Clipboard[i].hGlobalData );
            } else
                 <...den Benutzer warenen -
                     zu wenig Speicher...>
            break;
         case CF_BITMAP : /* Fragt nach Bitmap-Daten */
            Clipboard[i].hGlobalData =
                  CreateBitmap( 50, 50, 1, 1, NULL );
            if ( Clipboard[i].hGlobalData ) {
               <...Bitmap-Bits definieren...>
            } else
               <...den Benutzer warnen -
                   zu wenig Speicher...>
            break;
         case CF_METAFILEPICT :
            /* Fragt nach Metadatei-Daten */
            Clipboard[i].hGlobalData = GlobalAlloc( GHND,
                          (DWORD)sizeof(METAFILEPICT) );
            if ( Clipboard[i].hGlobalData ) {
               lpMetaFile = (LPMETAFILEPICT)GlobalLock(
                             Clipboard[i].hGlobalData );
               lpMetaFile->mm = MM_ANISOTROPIC;
               lpMetaFile->xExt = 0;
               lpMetaFile->yExt = 0;
               lpMetaFile->hMF = <...Metadatei definieren
                             und initialisieren...>
               GlobalUnlock( Clipboard[i].hGlobalData );
            } else
               <...den Benutzer warnen - zu wenig Speicher...>
            break;
         }
         /* Kopiert Daten in die Zwischenablage */
         SetClipboardData( wParam, Clipboard[i].hGlobalData );
      } else
           <...den Benutzer warnen -
               Anfrage wurde nicht beantwortet...>
      break;
   case WM_RENDERALLFORMATS :
      /* Darstellung aller zugesagten Formate */
      /* Öffnet die Zwischenablage */
      if ( OpenClipboard(hWnd) ) {
         for ( i=0; i<wFormats; i++ )
            if ( Clipboard[i].hGlobalData == NULL ) {
               /* Definiert Daten */
               switch( Clipboard[i].wFormat )
                  {
               case CF_TEXT : /* Fragt nach Textdaten */
                  Clipboard[i].hGlobalData = GlobalAlloc( GHND,
                                             (DWORD)100 );
                  if ( Clipboard[i].hGlobalData ) {
                     lpMemory = GlobalLock(
                                 Clipboard[i].hGlobalData );
                     lstrcpy( lpMemory,
                        "Some sample text for the clipboard." );
                     GlobalUnlock( Clipboard[i].hGlobalData );
                  } else
                     <...den Benutzer warnen -
                        zu wenig Speicher...>
                  break;
               case CF_BITMAP : /* Fragt nach Bitmap-Daten */

                  Clipboard[i].hGlobalData =
                     CreateBitmap( 50, 50, 1, 1, NULL );
                  if ( Clipboard[i].hGlobalData ) {
                     <...Bitmap-Bits definieren...>
                  } else
                     <...den Benutzer warnen -
                        zu wenig Speicher...>
                  break;
               case CF_METAFILEPICT :
                  /* Fragt nach Meta-Datei Daten */
                  Clipboard[i].hGlobalData =
                     GlobalAlloc(GHND, (DWORD)
                     sizeof(METAFILEPICT));
                  if ( Clipboard[i].hGlobalData ) {
                     lpMetaFile = (LPMETAFILEPICT)GlobalLock(
                                   Clipboard[i].hGlobalData );
                     lpMetaFile->mm = MM_ANISOTROPIC;
                     lpMetaFile->xExt = 0;
                     lpMetaFile->yExt = 0;
                     lpMetaFile->hMF = <...eine Metadatei
                                   definieren und sie
                                   initialisieren...>
                     GlobalUnlock( Clipboard[i].hGlobalData );
                  } else
                     <...den Benutzer warnen -
                        zu wenig Speicher...>
                  break;
               }
               /* Kopiert Daten in die Zwischenablage */
               SetClipboardData( Clipboard[i].wFormat,
                                 Clipboard[i].hGlobalData );
            }
      } else
         <...den Benutzer warnen -
             Zwischenablage kann nicht geöffnet werden...>
      break;
   default : /* eine andere Nachricht wird weitergegeben */
      return( DefWindowProc( hWnd, wMessage, wParam, lParam ) );
      break;
   }
   /* Normale Rückgabe */
   return( 0L );
}
```

Listing 4: *Dieser Programmausschnitt zeigt Code, der die Formatanfragen bei Gebrauch des verzögerten Wiedergabe-Schemas bearbeitet.*

Rich Text. Das Rich Text Format (RTF), ein relativ neuer Begriff von Windows, wurde von Microsoft als eine Methode zur Codierung von formatiertem Text und Grafik und zum einfachen Datentransfer zwischen Programmen entwickelt. Durch RTF können Texte und Grafiken zwischen Anwendungen, Umgebungen oder sogar Betriebssystemen ausgetauscht werden.

Ein RTF-Objekt besteht aus unformatierten 7-Bit ASCII-Text, mit speziell formatierten Befehlen im Dokument. Diese Kontrollwörter definieren Druckbefehle und liefern Informationen zur Verwaltung des Dokuments.

Inserentenverzeichnis

In die Zwischenablage kopieren

Die grundsätzliche Methode zum Kopieren von Daten in die Zwischenablage ist die Zuweisung eines globalen Speicherbereichs, die Formatierung des Speicherbereichs mit den gewünschten Daten und der Transfer der Speicher-Handle zur Zwischenablage. Es gibt zwei Modelle zum Kopieren in die Zwischenablage: das Ersetzen und das Hinzufügen. Das Ersetzen erfordert, daß die Zwischenablage geleert wird, bevor Daten hineinkopiert werden, um sicherzustellen, daß nur Daten aus dieser Kopieroperation in der Zwischenablage sind. Das Hinzufügen erfordert, daß beim Kopieren in die Zwischenablage diese nicht geleert wird. Vielmehr sollten nur Daten desselben Formats wie die Kopierdaten ersetzt, und alle anderen Formate unverändert übernommen werden. In fast allen Fällen ist das Ersetzen vorzuziehen, denn es benötigt weniger Speicher und ist im Konzept weit einfacher. Es gibt jedoch Fälle, wo das Hinzufügen angebracht ist; einer davon wird weiter unten demonstriert.

Die folgenden Programmausschnitte und die in den *Listings 1, 2, 3 und 4* zeigen, wie mehrfach dargestellte Daten in die Zwischenablage kopiert werden, und zwar unter Verwendung von sowohl normalen als auch von verzögerten Darstellungverfahren. Man beachte, daß die Kopierroutine das Ersetzen verwendet, und `EmptyClipboard` vor `SetClipboardData` aufgerufen wird. Das Ersetzen muß verwendet werden, wenn Daten mit einem verzögerten Darstellungverfahren übergeben werden, weil Windows die Meldung `WM_RENDERFORMAT` an den Zwischenablage-Besitzer sendet, wenn ein bestimmtes Format angefordert wird. Das Hinzufügen, das ohne `EmptyClipboard` arbeitet, überträgt kein Besitzrecht an die Zwischenablage. Wenn es in einer solchen Situation verwendet würde, könnten die Meldungen über die Format-Darstellung, die die Daten anfordern, eventuell nicht empfangen werden.

Sowohl die Kopier- als auch die Einfüge-Programmausschnitte verwenden die folgenden Datenstrukturen:

```
/* Datentabelle Zwischenablage-Format   */
typedef struct {
   WORD      wFormat;
   HANDLE    hGlobalData;
} CLIPBOARD;

/* Globale Anwendungsdaten */
WORD         wFormats;
CLIPBOARD    Clipboard[6];
```

Der Programmausschnitt in Listing 1 definiert drei verschiedene globale Datenblöcke, die zum Kopieren einer Mehrfach-Darstellung in die Zwischenablage verwendet werden. Wenn im anderen Fall ein verzögertes Darstellungschema gewünscht wird, können die Operationen für die Datenformatierung zurückgestellt werden, bis eine Anfrage über die Information empfangen wird, wie in Listing 2 gezeigt.

Wenn die Daten einmal definiert sind kann die Kopieroperation unter Verwendung des Programmausschnitts in Listing 3 durchgeführt werden. Zu beachten ist, daß Windows automatisch die gelieferten Datenblöcke neu zuweist und die verwendeten Speicher-Optionen verändert, um die `GMEM_DDESHARE`-Flags einzubinden.

Wenn ein verzögertes Darstellung-Schema verwendet wird, müssen die zugesagten Daten zu einer unbestimmten Zeit in der Zukunft geliefert werden. Obgleich die tatsächlichen Einzelheiten zur Erzeugung einer Datei ziemlich kompliziert sein können, ist der Code, der diese Anfrage liefert, einfach. Die Haupt-Fensterfunktion in Listing 4 zeigt, wie eine solche Formatanfrage abgearbeitet wird.

Das Einfügen

Daten aus der Zwischenablage einzufügen ist normalerweise etwas leichter als die entsprechende Kopierfunktion. Die generelle Methode zum Einfügen der Daten in die Zwischenablage sieht wie folgt aus:

- Öffne die Zwischenablage
- Entscheide, ob die Zwischenablage die gewünschte Darstellung hat
- Hole die Handle auf das Objekt, die Zwischenablage, oder die GDI-Objekt-Handle
- Lege eine lokale Kopie der Daten an
- Schließe die Zwischenablage

Unter Verwendung der oben definierten Datenstruktur zeigt das *Listing 5*, wie die Einfügeoperation erreicht wird, wenn man die Strukturen verwendet, die für die Listings 1, 2, 3 und 4 eingeführt wurden.

In Listing 5 wird `EnumClipboardFormats` solange aufgerufen, bis alle verfügbaren Formate durchgezählt sind. `IsClipboardFormatAvailable` könnte ebenfalls wiederholt aufgerufen werden, mit dem gewünschten Format als Parameter. Diese Funktion liefert `TRUE`, wenn das gefragte Format vorhanden ist, ansonsten `FALSE`. Die Zwischenablage muß nicht geöffnet werden, um diese Funktion zu nutzen.

Ein weiterer Punkt, der beachtet werden muß, ist die Art, in der die Handle, die von `GetClipboardData` zurückgeliefert wird, behandelt werden muß. Aufgrund der gemeinsamen Natur der Zwischenablage-Daten ist die Handle in Besitz von Windows, und die Daten, die sie repräsentiert, dürfen nicht verändert oder freigegeben werden. Des weiteren wird diese Handle ungültig, sobald `CloseClipboard` aufgerufen wird.

```
/* Lokale Variablen */
WORD        wCrntFormat;

if ( OpenClipboard(hWnd) ) {

   /* Initialisierung */
   wFormats = Ø;
   wCrntFormat = EnumClipboardFormats(Ø);
   /* Zählt verfügbare Formate durch */
   while ( wCrntFormat ) {

      /* Paßt Format, wenn gewünscht, ein */
      switch( wCrntFormat )
         {


      case CF_TEXT : /* Text verfügbar - Annahme < 64kb */
         {
            LPSTR       lpSrce;
            LPSTR       lpDest;
            HANDLE      hSrceText;
            HANDLE      hDestText;

            hSrceText = GetClipboardData( CF_TEXT );
            if ( hSrceText ) {
               hDestText = GlobalAlloc( GHND,
                  GlobalSize(hSrceText) );
               if ( hDestText ) {

                  /* Dupliziert Textblock */
                  lpSrce = GlobalLock( hSrceText );
                  lpDest = GlobalLock( hDestText );
                  lstrcpy( lpDest, lpSrce );
                  GlobalUnlock( hDestText );
                  GlobalUnlock( hSrceText );

                  /* Fügt Block an Format-Liste an */
                  Clipboard[wFormats].wFormat = CF_TEXT;
                  Clipboard[wFormats++].hGlobalData =
                     hDestText;

               } else
                  <...den Benutzer warnen -
                     zu wenige Speicher zum Einfügen...>
            } else
                  <...den Benutzer warnen -
                     Text kann nicht eingefügt werden...>
         }
         break;


      case CF_BITMAP : /* Bitmap ist verfügbar */
         {
            HDC      hDC;
            HDC      hSrceDC;
            HDC      hDestDC;
            HBITMAP  hSrceBitmap;
            HBITMAP  hDestBitmap;
            HBITMAP  hOldSrceBitmap;
            HBITMAP  hOldDestBitmap;

            hSrceBitmap = GetClipboardData( CF_BITMAP );
            if ( hSrceBitmap && GetObject(hSrceBitmap,
               sizeof(BITMAP),(LPSTR)&Bitmap) ) {

               /* Definiert zwei Speicherdarstellungen */
               hDC = GetDC( hWnd );
               hSrceDC = CreateCompatibleDC( hDC );
               hDestDC = CreateCompatibleDC( hDC );
               ReleaseDC( hWnd, hDC );

               if ( hSrceDC && hDestDC ) {
                  hDestBitmap =
                     CreateBitmapIndirect( &Bitmap );
                  if ( hDestBitmap ) {
                     /* Dupliziert Bitmap */
                     hOldSrceBitmap =
                        SelectObject( hSrceDC,
                                      hSrceBitmap );
                     hOldDestBitmap =
                        SelectObject(hDestDC,hDestBitmap);
                     BitBlt(hDestDC, Ø, Ø, Bitmap.bmWidth,
                        Bitmap.bmHeight, hSrceDC, Ø, Ø,
                        SRCCOPY);
                     SelectObject(hDestDC,hOldDestBitmap);
                     SelectObject(hSrceDC,hOldSrceBitmap);
                     /* Fügt Bitmap an Daten-Liste an */
                     Clipboard[wFormats].wFormat =
                        CF_BITMAP;
                     Clipboard[wFormats++].hGlobalData =
                        hDestBitmap;
                  } else
                     <...den Benutzer warnen -
                        zu wenig Speicher zum Einfügen...>
               } else
                  <...den Benutzer warnen -
                   Bitmap kann nicht eingefügt werden...>

               /* Löscht Speicherdarstellungen */
               if ( hDestDC )
                  DeleteDC( hDestDC );
               if ( hSrceDC )
                  DeleteDC( hSrceDC );

            } else
               <...den Benutzer warnen -
                  Bitmap kann nicht eingefügt werden...>

         }
         break;


      case CF_METAFILEPICT :
         /* Metadatei ist verfügbar */
         {
            HANDLE            hSrceMF;
            HANDLE            hDestMF;
            LPMETAFILEPICT    lpDestMF;
            LPMETAFILEPICT    lpSrceMF;

            hSrceMF = GetClipboardData( CF_METAFILEPICT );
            if ( hSrceMF ) {
               hDestMF = GlobalAlloc( GHND,
                     (DWORD)sizeof(METAFILEPICT) );
               if ( hDestMF ) {

                  /* dupliziert Metadatei */
                  lpSrceMF = (LPMETAFILEPICT)
                             GlobalLock( hSrceMF );
                  lpDestMF = (LPMETAFILEPICT)
                             GlobalLock( hDestMF );

                  lpDestMF->mm = lpSrceMF->mm;
                  lpDestMF->xExt = lpSrceMF->xExt;
                  lpDestMF->yExt = lpSrceMF->yExt;

                  lpDestMF->hMF = CopyMetaFile(
                                 lpSrceMF->hMF,
                                 (LPSTR)NULL );

                  /* Fügt Block in Format-Liste an */
                  if ( lpDestMF->hMF ) {
                     GlobalUnlock( hDestMF );
                     Clipboard[wFormats].wFormat =
                        CF_METAFILEPICT;
                     Clipboard[wFormats++].hGlobalData =
                        hDestMF;
                  } else {
                     GlobalUnlock( hDestMF );
                     GlobalFree( hDestMF );
                   <...den Benutzer warnen -
                  Metadatei kann nicht dupliziert werden...>
                  }

                  GlobalUnlock( hSrceMF );

               } else
                <...den Benutzer warnen -
                    zu wenig Speicher zum Einfügen...>
```

```
            } else
             <...den Benutzer warnen -
                 Metadatei kann nicht eingefügt werden...>
         }
         break;

      default : /* Etwas anderes - Ignorieren */
         break;
      }
      /* nächstes Format feststellen */
      wCrntFormat = EnumClipboardFormats( wCrntFormat );
   }

   /* Aufräumen */
   CloseClipboard();

} else
  <...den Benutzer warnen -
     Zwischenablage kann nicht geöffnet werden...>
```

Listing 5: *Programmausschnitt einer typischen Einfüge-Operation.*

```
LONG FAR PASCAL ViewerWndProc {
     HWND    hWnd;
     WORD    wMessage;
     WORD    wParam;
     LONG    lParam;
  }

{
     /* globale Variablen */
     static HWND  hNextWnd;
     /* Meldungsverarbeitung */
     switch (wMessage)
        {
     case WM_CREATE:
         /* Fenster wird angelegt */
         hNextWnd = SetClipboardViewer(hWnd);
         break;

     case WM_DRAWCLIPBOARD:
         /*Inhalt der Zwischenablage verändert */
         if (hnextWnd)
             SendMessage(hNextWnd, wMessage, wParam, lParam);
             <...ihre eigene weitere Verarbeitung hier...>
             break;

     case WM_CHANGECBCHAIN:
         /* Betrachterkette ändert sich */
         if (hNextWnd == (HWND)wParam)
             hNextWnd = LOWORD(lParam);
         else
             SendMessage(hNextWnd, wMessage, wParam, lParam);
         break;

     case WM_DESTROY:
         /* Fenster wird entfernt */
         ChangeClipboardChain(hWnd, hNextWnd);
         PostQuitMessage(0);
         break;

     default:
         return(DefWindowProc(hWnd, wMessage, wParam, lParam);
         break;
     }
     return(0L);
}
```

Listing 6: *Dieser Programmausschnitt zeigt, wie ein Fenster in die Zwischenablage-Betrachterkette eingebunden und wieder daraus entfernt werden kann.*

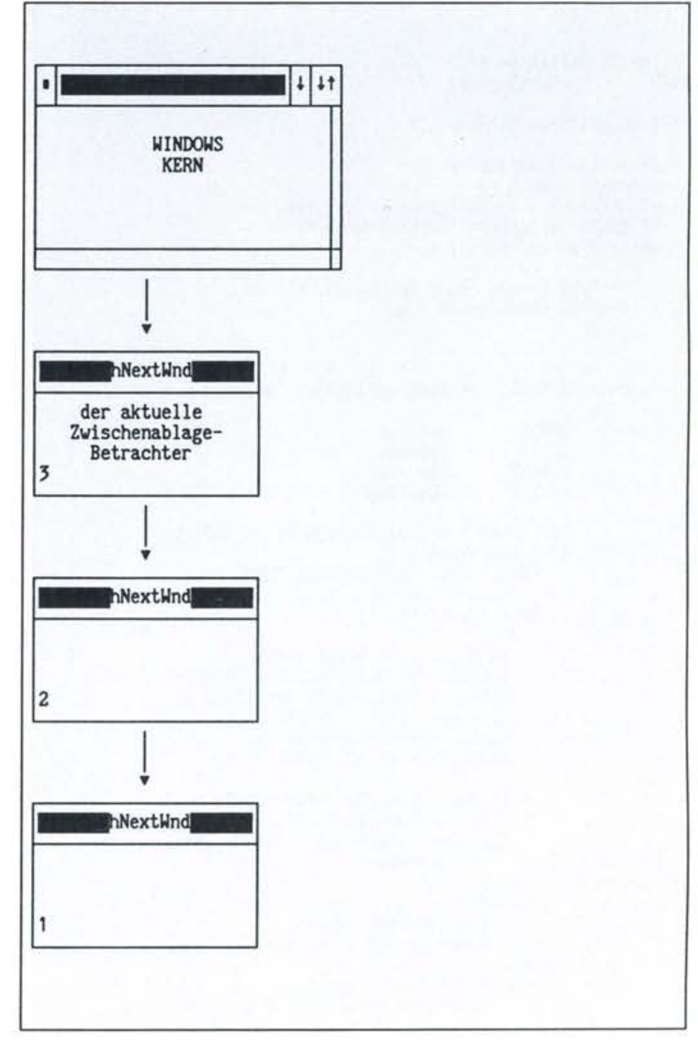


Bild 3: *Die Zwischenablage-Betrachterkette.*

Die Zwischenablage-Betrachterkette

Wie bereits erwähnt, bietet Windows einen Mechanismus zur Information interessierter Anwendungen, sobald Veränderungen in der Zwischenablage auftreten. Diese verkettete Liste von Windows-Handles, die Zwischenablage-Betrachterkette genannt wird, wird von jeder der beteiligten Anwendungen in einer kooperativen Weise gepflegt.

Eine Anwendung gelangt in diese Kette durch Aufruf der Funktion `SetClipboardViewer` mit ihrer Fenster-Handle als Parameter. Als Reaktion definiert Windows dieses Fenster als aktuellen Zwischenablage-Betrachter und liefert eine Handle zum vorherigen Betrachter zurück, wenn einer existiert. Die beteiligte Anwendung muß diese Handle speichern, da sie die einzige Verbindung zum Rest der Kette darstellt. Die Kette ist in umgekehrter Reihenfolge, mit dem neuesten Eintrag am Anfang der Kette, aufgebaut. *Bild 4* zeigt einen neuen Eintrag in der Betrachterkette.

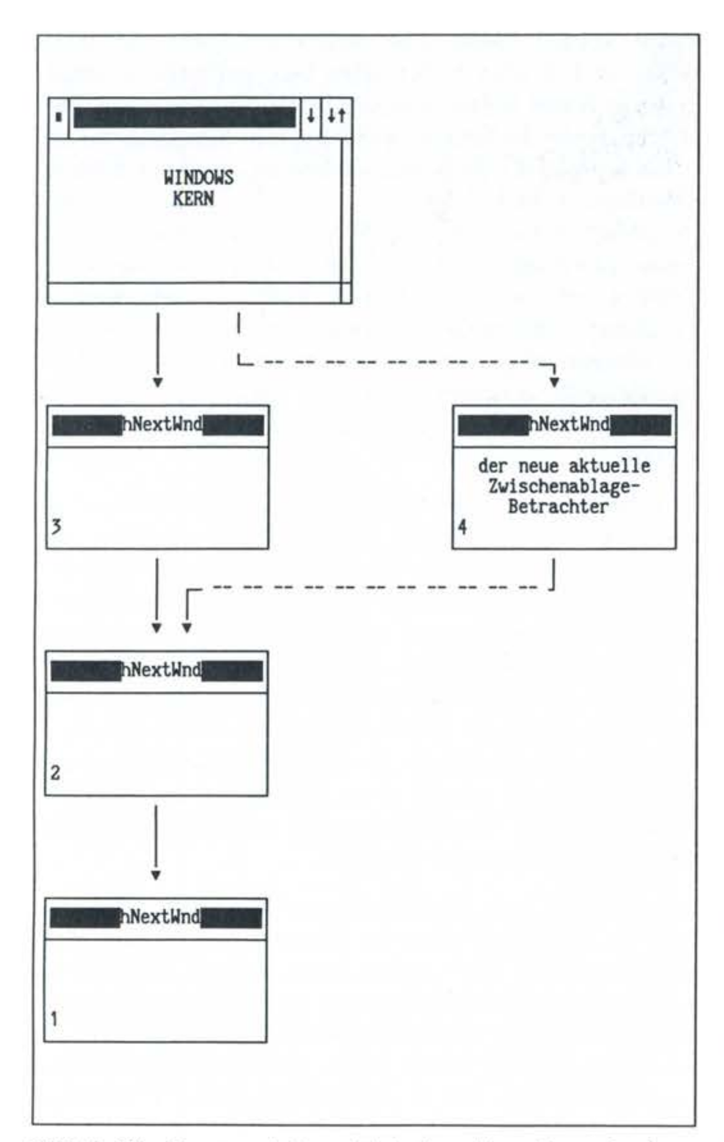


Bild 4: *Ein Fenster wird zur Zwischenablage-Betrachterkette hinzugefügt.*

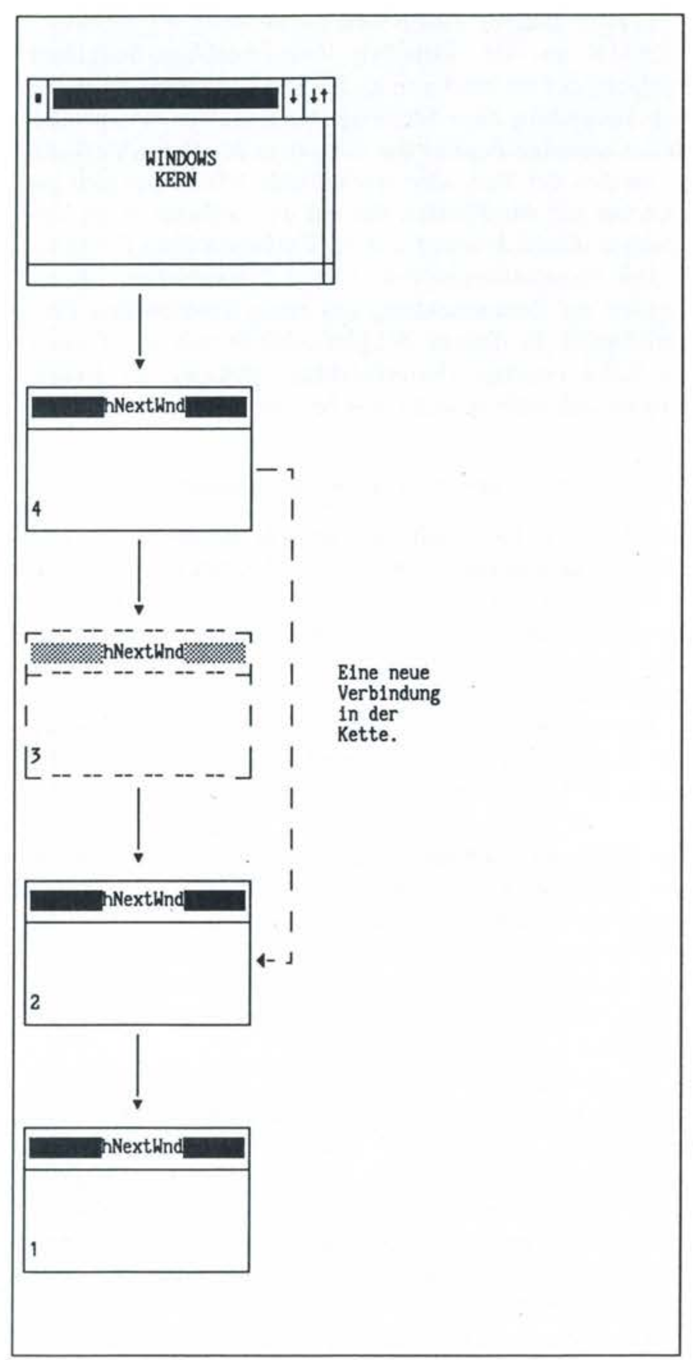


Bild 5: *Das Entfernen eines Fensters aus der Zwischenablage-Betrachterkette.*

Wenn Veränderungen der Zwischenablage auftreten, wird die Meldung `WM_DRAWCLIPBOARD` zum gerade aktuellen Betrachter gesendet. Dieses Fenster ist verantwortlich dafür, daß die Meldung durch einen expliziten `SendMessage`-Aufruf durch die Kette weiter nach unten weitergeben wird und danach alle für notwendig betrachteten Aktionen durchgeführt werden. Das Nichtweitergeben dieser Meldung durch die Kette zerstört die Kette und führt manchmal zu abnormem Verhalten von jenen Anwendungen weiter unten. Man beachte, daß die Meldung `WM_DRAWCLIPBOARD` während des Aufrufs von `CloseClipboard` erzeugt wird, das einen oder mehreren Aufrufen von `SetClipboardData` folgt, und daß die Kontrolle solange verloren bleibt, bis die gesamte Kette die Meldung bearbeitet hat.

Eine Anwendung kann sich aus der Betrachterkette durch Aufruf der Funktion `ChangeClipboardChain` abmelden. Dabei werden ihre Handle und die Handle der nächsten Anwendung in der Kette als Parameter übergeben. Wenn das fragliche Fenster der aktuelle Zwischenablage-Betrachter ist, setzt Windows einfach den aktuellen Betrachter in das nächste Fenster in der Liste. Dies entspricht dem Entfernen des ersten Eintrags aus einer verketteten Liste.

In allen anderen Fällen wird die Meldung WM_CHANGECBCHAIN an den aktuellen Zwischenablage-Betrachter geschickt, der sie die Kette nach unten weiterschickt. Wenn eine Anwendung diese Meldung erhält, muß sie überprüfen, ob das entfernte Fenster das nächste in der Betrachterkette ist. Ist dies der Fall, wird die Variable hNextWnd neu gesetzt, um auf das Fenster, das auf das entfernte folgt, hinzuweisen. *Bild 5* demonstriert das Entfernen eines Fensters.

Der Programmausschnitt *Listing 6* demonstriert dieses Konzept im Zusammenhang mit einer traditionellen Fensterfunktion. In diesem Beispiel schließt sich das Fenster der Zwischenablage-Betrachterkette während der Erzeugung an und entfernt sich selbst bei seiner Zerstörung.

Tips, Tricks und Problemlösungen

Der effektive Gebrauch der Zwischenablage kann eine Herausforderung für jeden Software-Entwickler sein, speziell wenn große und komplizierte Formate auftreten. Die Zwischenablage-Unterstützung wird sogar noch schwieriger, wenn einige große Programme um begrenzte Speicher-Ressourcen kämpfen.

Beim Gebrauch der Zwischenablage kann eine Anzahl von Techniken den ganzen Ablauf effektiver machen. Hier sind einige Programmiertips, -tricks und Problemlösungen.

Tip: Begrenzte Codesegmentgröße. Wenn eine oder mehrere große Anwendungen um die Speicher-Ressourcen kämpfen, verschlechtert sich die Systemleistung rapide; das gilt ganz besonders für die Arbeit mit Zwischenablage. Dies kann dadurch verbessert werden, daß bei der Entwicklung ihrer Anwendungen auf die Art der Speicherverwaltung von Windows Rücksicht genommen wird.

Im einzelnen können dramatische Verbesserungen der Leistung dadurch erzielt werden, daß jedes einzelne Codesegment als verschiebbar (moveable) und entfernbar (discardable) markiert wird, während seine Größe auf 8 Kbyte begrenzt wird. Danach kann Windows den Speicherbereich leicht neu einteilen, wenn Ressourcen knapp werden. Diese Neueinteilung funktioniert viel leichter wenn Codesegmente einheitlich klein sind.

Man sollte jedoch darauf aufpassen, daß die gewünschte Codesegmentgröße von 8 Kbyte auch auf das Codesegment TEXT angewandt wird. Dieses Segment enthält gewöhnlich einen großen Teil der C-Run-Time-Library. Um die Größe dieses Segments effektiv zu verringern, werden Sie den Quellcode der Run-Time Library erwerben müssen und ihn aufgeteilt in verschiedene Segmente, abhängig von Ihren Ansprüchen, neu kompilieren müssen.

Trick: Eingebaute Fehlerbehandlungs-Techniken. Viele der Programmausschnitte, die diesen Artikel begleiten, verwenden wenig ideale Fehlerbehandlungstechniken. Wenn man mit der Zwischenablage arbeitet, ist es besonders wichtig einige leistungsfähige Routinen einzubauen, die aufgerufen werden können, wenn eine bestimmte Operation fehlschlägt. In fast allen Fällen wäre man gut beraten, diese bereits in einem frühen Stadium in die Anwendungen einzubauen, bevor die Fehlerbehandlung eine Kernfrage wird.

Ein Beispiel für die Notwendigkeit einer solchen Fehlerbehandlungstechnik zeigt sich, wenn Daten in die Zwischenablage kopiert werden. Wenn die Intialisierung der Speicherzuweisung und die Speicherformatierung fehlschlägt, könnte eine alternative Methode verwendet werden um einigen entfernbaren Speicherplatz freizugeben und eine weniger anspruchsvolle und vielleicht weniger effiziente Methode zu verwenden. Der Gebrauch solcher Techniken unterscheidet große Anwendungen von den mittelmäßigen.

Problem: Der Gebrauch von festen Code- oder Datensegmenten. Der Gebrauch von festen Code- oder Datensegmenten kann die Windows-Speicherverwaltung beeinträchtigen. Programme, die solche Techniken verwenden, riskieren, daß sie den Einsatz von anderen, sich gut verhaltenden Programmen ausschließen. Obgleich Windows versucht, feste Code- oder Datensegmente abseits von kritischen Bereichen zu plazieren, gibt es Situationen, die schwerwiegende Verschlechterungen in der Systemleistung verursachen. Das wird speziell dann sichtbar, wenn die Zwischenablage zum Übertragen großer aneinanderliegender Datenelemente herangezogen wird.

Tip: Prüfung von Rückgabewerten und Nebeneffekten. Viele der API-Funktionen in Windows liefern nach der Ausführung einen Wert zurück. In den meisten Fällen ist es nicht ratsam, diese Rückgabewerte zu ignorieren, da ungewöhnliche Zustände entstehen können, wenn eine Funktion fehlschlägt. Darüber hinaus haben viele Windows-Funktionen Nebeneffekte, die zu unvorhergesehenen Resultaten führen. Des weiteren ist es leicht möglich, das der API-Code selbst sich mit einer neuen Version von Windows ändert, was zu neuen und unvorhersagbaren Situationen führen kann.

Trick: Fehlermeldungen stapeln. Wenn man die Zwischenablage verwendet, kann es zur Belegung des gesamten Systemspeichers kommen. In einer solchen Situation wird fast jede kompliziertere Operation fehlschlagen und möglicherweise in einer maschinengewehrartigen Abfolge von Meldungsfeldern ausarten.

Wo immer es möglich ist, ist es wünschenswert diese Meldungen zu stapeln und sie in einer einzigen Fehlermeldung zusammenzufassen. Der typische Anwender ist meistens nicht am exakten Grund eines auftretenden Fehlers interessiert; er kümmert sich mehr um den aktuellen Zustand der Anwendung und darum, was er oder sie machen kann, damit sie wieder läuft.

Problem: Unausführbarer Fehlerbehandlungscode. Ein Problem, auf das man sehr leicht stößt, tritt auf, wenn der Zustand eines Speicherüberlaufs erreicht wurde, der eine

komplexe Fehlerbehandlungsroutine erfordert. Wenn diese Routinen und deren dazugehörigen Ressourcen auch nur wenig Speicherbereich brauchen, sind sie ebenfalls leicht fehleranfällig. Das Resultat kann so günstig sein, daß es zu einer verstümmelten Fehlermeldung führt oder so fatal, das das System zusammenbricht. Generell sei gesagt, daß es ratsam ist, Fehlerbehandlungs-Code einfach zu gestalten, ihn, wann immer möglich, im Speicher zu halten und unabhängig von entfernbaren System-Ressourcen zu machen.

Tip: Vermeide lange Perioden von Inaktivität. Anwendungen, die mit der Zwischenablage arbeiten, brauchen manchmal viel Zeit, wenn intensive Berechnungen durchgeführt werden müssen. Ein Beispiel dafür ist das Anzeigen eines komprimierten Graustufen-TIFF-Bildes auf einem monochromen Bildschirm. In einem solchen Fall muß die Anwendung das Bild in bitweise Pixelzeilen dekomprimieren, während es die Graustufen in monochrom umwandelt. Wie leicht vorstellbar ist, kann diese Operation einen beträchtlichen Teil des Systems in Anspruch nehmen und einige Sekunden dauern.

In solchen Situationen wird am besten eine der folgenden Techniken verwendet:

`Kleine Jobs`	Verändern des Mauszeigers zu einer Sanduhr
`Mittlere Jobs`	Anzeige eines modalen Dialogfelds mit einer Abbruch-Schaltfläche
`Große Jobs`	Anzeige eines nicht-modalen Dialogfeldes mit einer Abbruch-Schaltfläche

Problem: Null-Darstellung-Fehler. Um den verzögerten Darstellungprozeß für andere Anwendungen transparent zu machen, muß der Zwischenablage-Besitzer alle Interaktionen korrekt ausführen. Wenn der Zwischenablage-Besitzer die Meldung `WM_RENDERFORMAT` oder die Meldung `WM_RENDERALLFORMATS` erhält, sollte er versuchen, die angeforderten Daten zu übergeben. Wenn er die Daten wegen ungenügendem Speicherplatz oder aus anderen Gründen nicht übergeben kann, sollte es der Zwischenablage-Besitzer unterlassen, die Funktion `SetClipboardData` mit einer Null-Handle aufzurufen. Dies würde zu einer falschen `WM_DRAWCLIPBOARD`-Meldung führen, die die Betrachterkette entlang nach unten gesendet wird, da Windows annimmt, daß die Zwischenablage sich wieder verändert hat, wobei in Wahrheit Null-Daten dargestellt werden. Wenn die Daten nicht dargestellt werden können, ist es ratsam, gar nichts zu tun. Das nächste Mal, wenn die Daten angefordert werden, wird Windows eine weitere `WM_RENDERFORMAT`-Meldung senden. Die Anfrage mag diesmal erfolgreich sein, da die exakte Speicherkonfiguration, zu der bei der letzten Anfrage verschieden sein könnte.

Problem: Gebrauch von verzögerter Darstellung bei Nicht-Bereitschaft. Beim Aufruf von `CloseClipboard` wird unmittelbar nach dem Kopieren von Daten in die Zwischenablage eine `WM_DRAWCLIPBOARD`-Meldung die Betrachterkette entlang nach unten gesendet. Wenn einer der Zwischenablage-Betrachter unmittelbar die Daten anfragt, wird eine `WM_RENDERFORMAT`-Meldung an den gegenwärtigen Zwischenablage-Besitzer gesendet. (Beachten Sie den Wiedereintritt, da die `CloseClipboard`-Funktion noch nicht abgeschlossen ist.)

Diese unmittelbare Reaktion von einer oder mehreren Anwendungen in der Betrachter-Kette kann auftreten, noch bevor die Kopierfunktion bereit ist, Daten zu senden. In diesem Fall wird eine inkorrekte Antwort an die `WM_RENDERFORMAT`-Meldung übergeben, wodurch die Zwischenablage in einem unbestimmbaren Zustand gerät, und eine Menge anderer Probleme verursacht wird. Dies kann dadurch vermieden werden, daß man `CloseClipboard` erst aufruft, wenn die Daten zufriedenstellend übergeben werden können.

Problem: Zusätzliche `WM_DRAWCLIPBOARD`-Meldungen. Ein weiteres Problem, dem man vorbeugen sollte, taucht auf, wenn ein Mitglied der Zwischenablage-Betrachterkette eine Anfrage für Null-Daten stellt. Beim Aufruf von `GetClipboardData` durch eine Anwendung sendet Windows eine `WM_RENDERFORMAT`-Meldung zum derzeitigen Zwischenablage-Besitzer. Hinter den Kulissen sendet Windows eine zusätzliche `WM_DRAWCLIPBOARD`-Meldung die Betrachterkette nach unten als Antwort auf den `SetClipboardData`-Aufruf. Die einfügende Anwendung empfängt diese falsche Meldung, da sie ebenfalls ein Mitglied der Kette ist und kann dazu verleitet werden, zu glauben, daß sich der Zwischenablage-Inhalt schon wieder verändert hat. Der einzige Weg dies zu verhindern ist das Führen eines Flags, das sinnvolle `WM_DRAWCLIPBOARD`-Meldungen von falschen unterscheidet.

Fazit

Obgleich das Einbinden guter Zwischenablage-Unterstützung in ein Programm herausfordernd und zeitaufwendig sein kann, ist es der Nutzen wert, den man daraus zieht. Da immer mehr Software unter Windows auf den Markt kommt, werden Produkte neben ihren Leistungen zunehmend nach ihren Verbindungsmöglichkeiten beurteilt. Umfassende Unterstützung der Zwischenablage ist ein natürlicher und logischer Weg jene interne Verknüpfung zu erreichen, die von Benutzern erwartet wird.

Trotz der Komplexität dieses Artikels wurden viele Punkte noch nicht aufgezeigt. Die komplizierteren Zwischenablage-Formate haben alle jeweils ihre eigenen Spezifikationen. Es ist einiges gutes Referenzmaterial verfügbar, welches das Programmieren der Zwischenablage detailliert erklärt. Das beste davon ist »Programming Windows« von Charles Petzold (Microsoft Press, 1988). Mit diesem und anderem Material lassen sich selbst die kompliziertesten Zwischenablage-Probleme mit etwas Geduld lösen.

Kevin P. Welch / David E. West

Ein Vorschlag für einen weiteren Windows-Grafikstandard:

Encapsulated Postscript

Encapsulated PostScript (EPS) wird immer stärker das Standardmittel zum Transfer von Grafiken und Text zwischen unterschiedlichen Anwendungen unter verschiedenen Systemumgebungen. Diese industrieweite Anerkennung des Standards führte zur Integration von EPS-Unterstützung in viele populäre Programme.

EPS wurde für die Microsoft Windows-Umgebung auf IBM-PC-kompatiblen enthusiastisch aufgenommen. Windows selbst hat sich als überaus konsistente und benutzerfreundliche Umgebung bewiesen, die die Entwicklung von grafikorientierten Anwendungen erleichtert. EPS ist von speziellem Interesse für solche Anwendungen, da es als geräteunabhängiges Mittel zum Transport grafischer Objekte zwischen Anwendungen und Umgebungen dienen kann.

Es existieren nun einige Anwendungen unter Windows, die diesen Standard unterstützen, und viele weitere sind in Entwicklung. Diese Anwendungen sind im Moment auf das Importieren und Exportieren von EPS-Dateien auf Diskette beschränkt, was normalerweise nur mit einem expliziten Benutzerbefehl ausgeführt werden kann und etwas mühsam zu handhaben ist. Auch verkompliziert das unnötig die integrierte, freundliche Umgebung von Windows.

Diese Situation würde sich durch die Definition eines Zwischenablage-Datenformates für den EPS-Standard entscheidend verbessern. Dadurch würde es ermöglicht, PostScript-Bilder über die Zwischenablage und durch Dynamic Data Exchange (DDE) zu übertragen. Anwendungen hätten dann gemeinsamen Zugriff auf Text und Grafiken, indem sie dieselben Mechanismen wie für andere vordefinierte Formate benutzen.

Anforderungen

Jede Definition eines neuen Zwischenablage-Datenformats für EPS muß verschiedene Entwurfskriterien und Anforderungen erfüllen, die der exakten Festlegung der Spezifikation dienen. Dazu gehören unter anderem die folgenden Kriterien:

1. Die Spezifikation sollte die Fähigkeiten der Zwischenablage zu Mehrfach-Darstellungen berücksichtigen und so viele Standard-Darstellungen der Bilder wie möglich für existierende Windows-Anwendungen anbieten. Dies würde solchen Programmen wie Write und Paint Zugriff auf Bildschirm- und Drucker-Darstellungen von PostScript-Bildern geben.
2. Die Spezifikation muß nicht unbedingt mit beliebig großen und komplexen Bildern arbeiten. Wie es bei dem `CF_TIFF`-Zwischenablage-Format der Fall ist, werden große Bilder vielleicht am besten durch einen anderen Mechanismus übergeben.
3. Der Transfer von kleinen Bildern zwischen Anwendungen (etwa zwischen 10 Kbyte und 64 Kbyte) sollte so effizient wie möglich durchgeführt werden und unter Verwendung von Standard-Zwischenablage-Operationen vollständig im Speicher ablaufen.
4. Die zukünftige Definition einer Bitmap-Maske für jede Bild-Darstellung, die als Teil dieser Spezifikation vorgesehen ist, sollte erlaubt werden. Dies ermöglicht der Ziel-Anwendung, nicht-rechteckige Bild-Darstellungen über existierende Grafiken und Texte zu lagern.
5. Es sollte eine Unterscheidung zwischen den PostScript-Texten und den verschiedenen Bild-Darstellungen, die als Teil des Standards vorhanden sind, gemacht werden. Dies würde mögliche zukünftige Systeme berücksichtigen, die Display-PostScript verwenden, und den Bedarf an speziellen Bildschirm-Darstellungen verringern.

Ferner ergibt sich aus einer Analyse von EPS-Grafiken auf IBM-kompatiblen PCs, daß die Bilder in einem Bereich von ca. 2 Kbyte bis über 3 Mbyte liegen. Die meisten der weniger komplizierten Bilder sind kleiner als 64 Kbyte und gehen daher mit den Entwicklungszielen dieser Spezifikation konform.

Spezifikations-Vorschlag

Der folgende Vorschlag für eine Spezifikation beschreibt die Mechanismen und Standards, die zum Transfer von EPS-Bildern via Zwischenablage dienen. Diese Diskussion arbeitet mit der vorhandenen EPS-Dateidefinition und berücksichtigt keine geräteabhängigen Veränderungen dieses Standards in der Zukunft.

1. EPSF-Objekte sind ein Zusammenschluß von Standard-PostScript-Text mit einer oder mehreren zusätzlichen Bild-Darstellungen, wie etwa eine Windows-Metadatei, ein TIFF-Bild oder möglicherweise eine Macintosh-PICT-Darstellung.
2. Jede Anwendung, die den Zwischenablage-Transfer von EPSF-Bildern unterstützt, müßte ein PostScript-Zwischenablage-Format registrieren. Dies würde dadurch erreicht, daß die folgende Zeichenkette an die Windows-Funktion `RegisterClipboardFormat` übergeben wird:
   ```
   Postscript
   ```
   Mehrfache Registrierung dieses Formats bewirkt die Rückgabe derselben ID-Nummer mit einem um eins erhöhten internen Referenz-Zähler. Mehrere kooperierende Anwendungen könnten sich dann dieselben Formatnummern teilen, auch wenn diese von Sitzung zu Sitzung variieren.
3. Eine Anwendung, die ein EPSF-Bild über die Zwischenablage überträgt, würde einen PostScript-Text mit einer oder mehreren Bilddarstellungn übergeben. Die liefernde Anwendung wäre verantwortlich für die Entscheidung, ob genügend Speicherplatz für die Kopieroperation vorhanden ist und welche der zusätzlichen Darstellungen geliefert werden.
   Die zusätzlichen Darstellungen wären normalerweise einfache Zwischenablage-Kopien der abgebildeten Dar-

stellungen, die mit dem PostScript-Text mitgeliefert werden. In einigen Situationen möchte eventuell die liefernde Anwendung zusätzliche Darstellungen der Bilder erzeugen. Zum Beispiel könnte eine Windows-Metadatei-Darstellung in einer Monochrom-Bitmap verarbeitet werden, wodurch Anwendungen wie Windows Paint Zugriff zu einer groben Darstellung der PostScript-Abbildung gewinnen würden. Die liefernde Anwendung wäre in diesem Beispiel für die Bestimmumg der Abmessungen dieser zusätzlichen Bitmap-Darstellung verantwortlich.

Es ist zu beachten, daß in den meisten Fällen die liefernde Anwendung die PostScript- und Bilddarstellungn auf einer verzögerten Basis liefern würde, damit die Generierung von ungewünschten Formaten und die Belegung von wertvollem Speicherplatz verhindert wird.

4. Das PostScript-Zwischenablage-Objekt bestünde aus Standard-PostScript-Text, der direkt aus der EPSF-Datei genommen wird. Anwendungen, die den PostScript-Text anzeigen wollen, können dies mit Standard-ASCII-Textbearbeitungs-Mechanismen erreichen.
5. In einigen Situationen wäre es wünschenswert, eine Shadow-Mask für jede Bitmap-Darstellung des EPSF-Bildes zu liefern. Das würde einfügenden Anwendungen das Überlagern der Bitmaps über vorhandenen Text und Grafiken erlauben. Trotz der Vorteile dieser Möglichkeit empfehlen wir, daß derzeit kein zusätzliches Format für diese Maske definiert wird.

   Zu beachten ist, daß viele CF_TIFF-Bilder, das des Adobe Illustrator™ eingeschlossen, eine TIFF-Maske als Teil der gesamten Abbildung beinhalten.
6. Die kopierende Anwendung wäre verantwortlich für die Freigabe jedes globalen Speicherbereichs, der mit dem PostScript-Text verbunden ist, wenn die Daten durch einen anderen Eintrag ersetzt werden, obgleich dies normalerweise Aufgabe von Windows selbst ist. Der Empfang der Meldung `WM_DESTROYCLIPBOARD` würde aufzeigen, das dies passiert ist. Die kopierende Anwendung, welche in diesem Schema nicht Teil der Zwischenablage-Betrachterkette sein muß, kann dann den Speicherplatz, der mit dem PostScript-Zwischenablage-Objekt verbunden ist, freigeben.

Implementierung

Diese Spezifikation ist, bis zu diesem Punkt, nur ein Vorschlag. Die Definition wird sich wahrscheinlich verändern, sobald alle interessierten Parteien ihren Kommentar abgegeben haben. Die endgültige Anerkennung und Implementation dieses Standards kann auf drei Arten erfolgen, die im folgenden, in absteigender Reihenfolge Ihrer Erwünschtheit, aufgezeigt werden.

Erstens könnte Microsoft ein neues Zwischenablage-Format, zum Beispiel `CF_POSTSCRIPT`, definieren und es zu einem Teil eines zukünftigen Windows Software Develpment Kit (SDK) machen. Dies würde eine aktualisierte Version von `WINDOWS.H` und eine überarbeitete Dokumentation erfordern. Microsoft würde dann der Pfleger der Definition und verantwortlich für seine Unterstützung und möglichen zukünftigen Erweiterungen sein. Die Windows-Zwischenablage wäre dann für die Freigabe von globalem Speicherplatz verantwortlich, der von PostScript-Text konsumiert wird, wodurch die liefernde Anwendung von dieser Aufgabe befreit wird. Des weiteren wäre sie verantworlich dafür, daß sich nicht unbenutzte Ressourcen häufen.

Zweitens: Der wahrscheinlichste Ablauf ist, daß sich eine repräsentative Gruppe von Software-Entwicklern, nachdem sie eine Übereinstimmung über die Spezifikation gefunden haben, die Definiton bekannt machen und tatsächlich in einigen sich ergänzenden Produkten implementieren. Jedes dieser Produkte wäre für die Registrierung des PostScript-Zwischenablage-Formats und für die jeweilige Unterstützung verantwortlich. Nach einiger Zeit würde das Zwischenablage-Protokoll zum Bearbeiten dieses Formats ein Industrie-Standard, der von allen anerkannt würde, bis es durch etwas Besseres ersetzt würde. Unter diesen Voraussetzungen wäre der Pfleger der Definition ein Komitee von Entwicklern, das für die Spezifikation verantwortlich wäre, bis diese die Berechtigung zur Einbeziehung in das Windows-SDK erworben hätte.

Die letzte und am wenigsten wünschenswerteste Art zur Etablierung eines Standards ist, daß eine kleine Gruppe von Software-Entwicklern eine Übereinstimmung über die Spezifikation findet, sie zum Teil ihrer Produkte machen, aber sie nicht der Öffentlichkeit zur Verfügung stellen. Diese würde vielleicht kurzzeitig diesen sich ergänzenden Programmen einen Marktvorteil geben bis der Standard veröffentlicht, oder durch einen anderen veröffentlichten ersetzt würde. Wie dem auch sei, von einem historischen Standpunkt betrachtet würde unter diesen Voraussetzungen der Standard wahrscheinlich kurzlebig sein, und niemals den Grad an Verbreitung erreichen, der zur Qualifizierung eines anerkannten Industriestandards nötig wäre.

Zukünftige Möglichkeiten

Die Entwicklung eines Mittels zum Transfer von EPS-Bildern über die Windows-Zwischenablage eröffnet einige interessante Möglichkeiten. Die Spezifikation und geplante Definition eines Standardmittels zum Erreichen dieser Aufgabe ist ein Versuch zur Lösung des begleitenden Problems und zur Realisierung der Spezifikation. Viele Anwendungen, die derzeit in Gebrauch oder Entwicklung sind, könnten diese Spezifikation ausnützen und darüber hinaus PostScript als die wünschenswerte Seitenbeschreibungssprache für Microsoft Windows bestätigen.

Dieser Artikel ist nicht das letzte Wort zu diesem Thema. Interessierte Entwickler sind aufgefordert, so schnell wie möglich auf diesen Artikel zu reagieren und detaillierte Vorschläge zu seiner Verbesserung einzubringen. Bitte richten Sie alle Kommmentare, Vorschläge oder Kritiken an die folgende Adresse:

Kevin P. Welch, President, Eikon Systems, Inc., 989 East Hillsdale Blvd., Suite 260, Foster City, CA 94404, (415) 349-4664

Windows/386-Batchtool Bridge/386:

Die Integration von Windows/386-Anwendungen

In den letzten 12 Monaten wurde von Softbridge Microsystems ein neuartiges Windows-Anwendungsprogramm entworfen. Ursprünglich nur ein Nebenprodukt hausinterner Entwicklungen, hat Bridge/386 seine Eigenständigkeit dadurch erreicht, daß es dem Benutzer erlaubt, die Windows-Umgebung nach eigenen Wünschen einzurichten.

Bis vor kurzem war Softbridge aus Cambridge, Mass., auf vertikale Anwendungen spezialisiert, unter anderem auf Software für Frachtgutverwaltung (The Positioning Decision Support System), Finanzplanung (The Softbridge Financial Planner), Versicherungs- (The Flint Expert System) und Bankwesen (The Softbridge Commercial Banker).

Während der Erstellung dieser Anwendungsprogramme haben Softbridge-Programmierer oft ihre eigenen System-Tools geschrieben, viele davon unter Microsoft Windows. Softbridge arbeitet seit März 1984 mit Windows und war damit einer der ersten unabhängigen Software-Hersteller, die es verwendeten. In den letzten vier Jahren hat Softbridge viel Erfahrung mit Windows gesammelt und dies brachte die Firma dazu, ihre Aufmerksamkeit über ihre traditionellen Produkte für den vertikalen Markt hinaus auf andere Programme zu richten.

Softbridge erkannte, daß Windows zwar über eine Reihe ausgezeichneter Präsentationsobjekte verfügt, wie Fenster, Bildrolleisten und Dialogfelder, aber dem Benutzer wenig Möglichkeiten bietet, auf diese Objekte zuzugreifen. Ein Programmierer kann diese Windows-Einrichtungen sicherlich ausschöpfen, der Benutzer jedoch kann sie nur verwenden, wenn es das jeweilige Anwendungsprogramm erlaubt. Zum Beispiel war es dem Benutzer bisher unmöglich, ein eigenes Dialogfeld zu entwerfen.

Unter DOS verfügt ein Benutzer über eine Stapelverarbeitungssprache, mit der sich, in begrenztem Rahmen, Menüs aufbauen oder wiederholende Abläufe automatisieren lassen. Unter Windows verliert der Benutzer sogar diese begrenzte Möglichkeit. Softbridge wollte dem Benutzer etwas von dieser Funktionalität zurückgeben und begann eine Anzahl von Tools zu entwickeln, mit welchen man die Windows-Umgebung nicht auf Programmier-, sondern auf Benutzerebene verwalten kann.

Softbridge arbeitete diese Idee bald weiter aus und begann ein System zu entwickeln, das dem Benutzer erlaubte, nicht nur Windows-Anwendungen, sondern auch Standardpakete wie Lotus 1-2-3 oder dBASE III zu integrieren. Was schließlich dabei entstand, war ein Tool, das dem Benutzer die Automatisierung der Ausführung sowohl Windows- als auch Windows-fremder Programme unter Windows erlaubt, auf ähnliche Weise wie es mit der DOS-Batchsprache möglich ist.

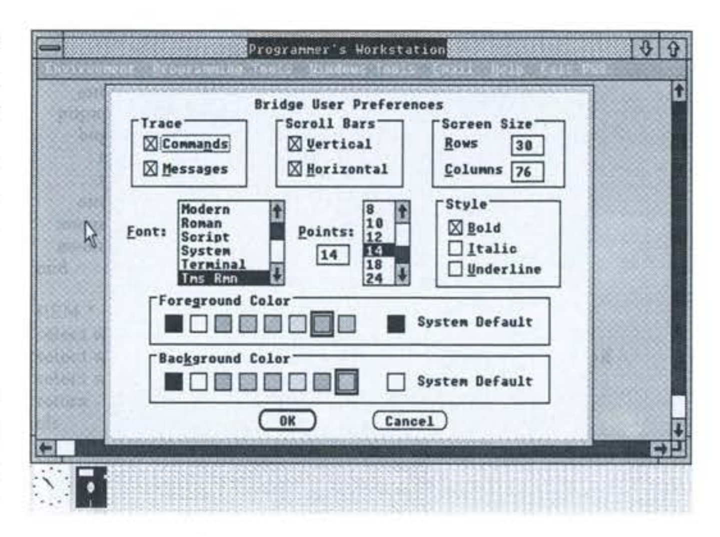


Bild 1: *Bridge/386 bietet dem Benutzer umfangreiche Möglichkeiten zur Anpassung des Bridge-Fensters.*

Das Ergebnis ist ein abgerundetes System, das es der EDV-Abteilung einer Firma beispielsweise gestattet, Anwendungsprogramme für Ihre Benutzer zu kontrollieren, zu verwalten und zu integrieren. Vielleicht noch wichtiger ist, daß diese Software die Integration von verschiedenen Softwareprodukten unter der gemeinsamen Benutzerschnittstelle von Windows ermöglicht. Sie gibt dem PC-Verwalter und auch dem Benutzer die Möglichkeit eigene Software zur Steuerung von Anwendungen zu entwickeln.

Als Microsoft die letzten Versionen von Windows, Windows/286 und Windows/386 vorstellte, konnte Softbridge bereits Bridge/286™ und Bridge/386™ ankünden. Darüber hinaus wird Softbridge im vierten Quartal 1988 Bridge/OS vorstellen, eine Version von Bridge für OS/2 und den Presentation Manager. Dies wird es ermöglichen, Arbeitsumgebungen, die unter Bridge und Windows/386 oder Windows/286 entwickelt wurden, leicht auf OS/2 umzustellen.

Im Gegensatz zu den vertikalen Anwendungsprogrammen von Softbridge sind die Produkte Bridge/286 und /386 Systemsoftware, mit welcher der Benutzer Windows-Anwendungen erweitern und entwickeln kann. Laut Fred Ciaramaglia, dem Vizepräsidenten der Abteilung Technologie, ist Softbridge nun daran interessiert, einen Ruf als Lieferant für Systemsoftware und nicht nur Anwendungssoftware zu erlangen. Dies bedeutet, daß Bridge, welches als internes Tool für Anwendungsentwicklungen anfing, nun als Systemprodukt über die üblichen Verkaufskanäle vertrieben wird.

Ein besonders interressanter Aspekt von Bridge ist, daß es unter jedem NetBIOS-kompatiblen LAN arbeitet, wodurch es Benutzeranwendungen transparent in einem Netzwerk vereinigen kann, während der Benutzer eine scheinbar lokale Arbeitsumgebung beibehält. PC-Verwalter können so dem Anwender eine Menge der Komplexität von Netzwerken abnehmen.

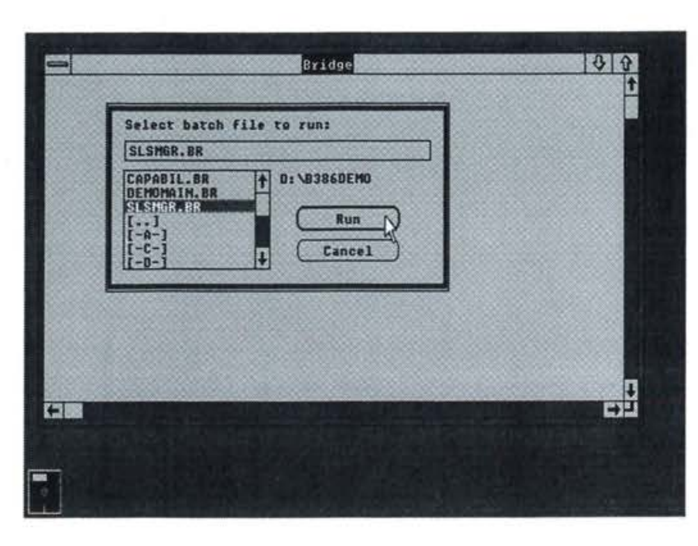


Bild 2: *Bridge startet als ganz normales Programm mit der Titelleiste »Bridge«.*

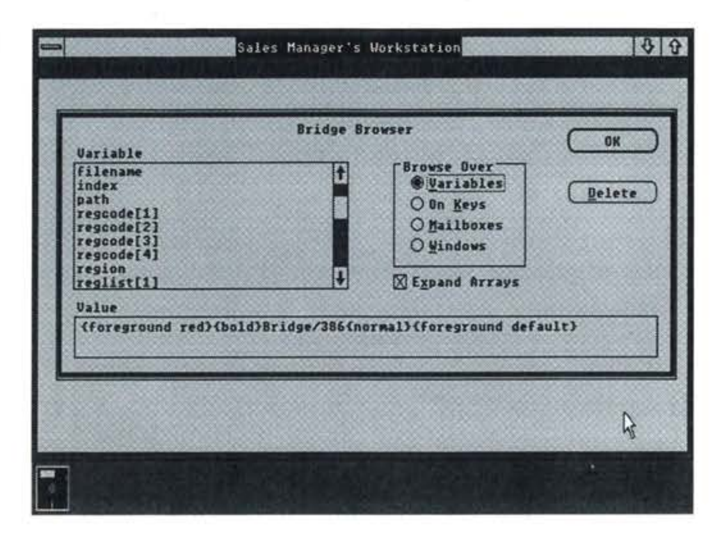


Bild 3: *Wenn der Benutzer ein Bridge-Batchprogramm startet (in diesem Fall* `SLSMGR.BR` *aus Bild 2), befindet sich das Bridge-Fenster weiterhin auf dem Bildschirm; die Titelzeile ändert sich jedoch, so daß es aussieht, als ob ein neues Fenster geöffnet worden ist.*

Die wichtigste Stärke von Bridge/386 ist jedoch seine Schnittstelle zu Windows, die Möglichkeit, einige der Windows-Tools direkt unter Kontrolle des Anwenders arbeiten zu lassen, und natürlich das Multitasking von Anwendungen auf 80386-Systemen.

Bridge/386

Bridge/386 wird am besten als speziell für Windows entwickelte Batchsprache bezeichnet. Es erlaubt dem Benutzer, wiederholende Windows-Abläufe zu automatisieren und sich mehrerer Anwendungsprogramme zu bedienen, ohne Windows zu verlassen. Ich habe Bridge/386 auf einem Intel InBoard 386 eingesetzt, um eine Windows-Anwendung aufzubauen, welche ich »The Integrated Programmer's Workstation« (PWS) nenne und später vorstellen werde.

Bridge/386 eröffnet Möglichkeiten, die denen gleichen, die mit der DOS-Batchsprache verfügbar sind. Es erweitert Windows um die folgenden Möglichkeiten:

- benutzerdefinierte Menüs
- benutzerdefinierte Dialogfelder
- verbesserte bedingte Verarbeitung
- Unterprogrammaufrufe
- Feldvariablen
- Meldungen zwischen Anwendungen
- Netzwerkfähigkeit
- »On«-Bedingungen (Verzweigungen und spezielle Abläufe, die durch eine zukünftige Bedingung auftreten)
- Parameter, die an Aufrufdateien übergeben werden können
- Unterstützung für benannte Arrays (verbessert die DOS Variablenmöglichkeiten `%1 - %9`)
- eine `ERROR`-Variable, eine umfassendere Version der DOS-Variable `ERRORLEVEL`
- die Fähigkeit, mehrere Befehle in eine Zeile zu setzen (durch Semikolon getrennt)

Wie bei den DOS-Batchfunktionen schreibt man mit Bridge/386 zuerst eine gewöhnliche ASCII-Textdatei, welche die Bridge/386-Befehle enthält. Anstelle einer `BAT`-Datei erzeugt man eine Datei mit der Erweiterung `BR`, wodurch die Datei als Bridge/386-Batchdatei gekennzeichnet wird. Von Windows/386 aus startet man dann `BRIDGE.EXE`, das Steuerprogramm von Bridge/386. `BRIDGE.EXE` ist ein Windows-Programm (*Bild 2*), das genauso aussieht wie jedes andere. Es besitzt sein eigenes Fenster und dient als Kontrollstelle für die zu entwickelnde Anwendung.

Das Bridge/386-Fenster kann so eingerichtet werden, daß es an jeder Stelle des Bildschirms erscheint, wo immer es der Anwender wünscht. Es kann jede Größe und Gestalt annehmen und die Bridge-Titelleiste zeigt an, was der Anwender angezeigt haben will (*Bild 3*). Diese Anpassungsmöglichkeit erlaubt dem Anwender, Bridge/386 während der Ausführung zu verbergen und den Anschein zu erwecken, daß der Benutzer sein eigenes Windows-Anwendungsprogramm verwendet. Durch Einbau einer Standardzeile wie `BR=Bridge.EXE ^.BR` in die Datei `WIN.INI` kann der Benutzer jede Anwendung, die die Erweiterung `BR` besitzt, aus dem MS-DOS-Fenster aufrufen, und es scheint dann ein Windows-Programm zu sein.

Es kann immer nur eine Bridge/386-Batchdatei ausgeführt werden, bei Aufruf von mehreren stellt Bridge deren Ausführung in eine Warteschlange bis vorher aufgerufene Batchdateien beendet sind.

Wenn Bridge zum ersten Mal gestartet wird, erscheint das Standard-Systemmenü in der oberen linken Ecke des Fensters (*Bild 4*). Dieses Menü erlaubt die direkte Eingabe von Bridge-Befehlen, beziehungsweise die Ausführung von Batchdateien.

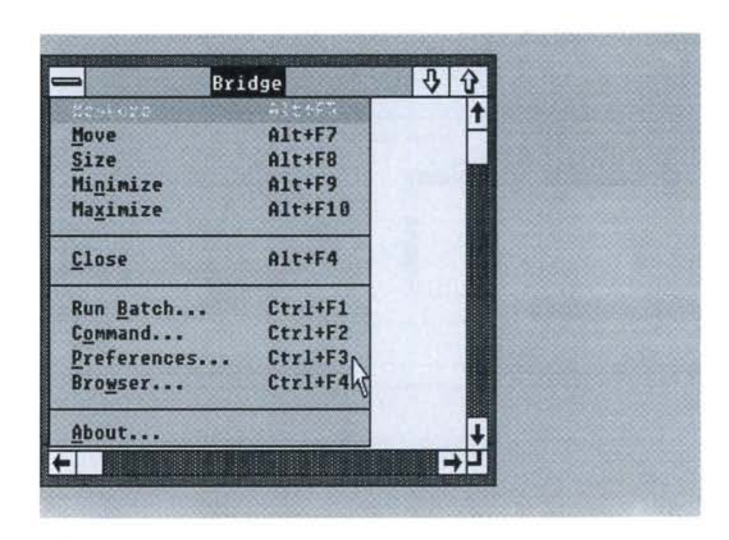


Bild 4: *Das System-Menü von Bridge.*

BRIDGE.EXE	Der Bridge-Kernel und die Bridge-Batchschnittstelle
BRDYN.EXE	APIs für andere Bridge-Programmierschnittstellen.
BRMSG.EXE	Der Bridge-Message-Manager.
BRDOS.EXE	Der Bridge-DOS-Supervisor und die DOS-Batchschnittstelle.
BR.COM	Dient zur Ausgabe von Befehlen an den Bridge-Kernel, während man mit DOS arbeitet. Befehle können auch von Programmen aus über eine Interrupt 16H-Schnittstelle gegeben werden.

Tabelle 1: *Die Bestandteile von Bridge*

Zwei wichtige Optionen in diesem Menü sind `Browser` und `Preferences`. `Browser` ermöglicht die Prüfung und Steuerung des derzeitigen Bridge-Prozesses. `Preference` wird zur Auswahl von Optionen verwendet, zum Beispiel Farbe, Bildlaufleisten, Bridge-Schriftart und ob die Trace-Ausgabe von ausführenden Batchdateien an das Bridge-Fenster gesendet werden soll. Ich werde den `Browser` näher erklären, wenn ich zeige, wie ich ihn zum Testen und Debuggen von PWS verwendet habe.

Die Teile von Bridge

Bridge besitzt zwei Hauptteile: Die »Bridge Engine« und die verschiedenen APIs (Anwendungsprogrammierschnittstellen), die den Zugriff zu dieser »Engine« ermöglichen. Der »Kernel« stellt den Hauptteil der Engine dar. Er kann Anwendungen öffnen, aktivieren und schließen sowie Tastatureingaben zwischen Anwendungen abfangen und abarbeiten. Der Kernel verwaltet auch die Menüs und Dialoge, die zum Aufbau von Benutzerschnittstellen in Bridge-Programmen verwendet werden können. Bedingungen wie `ON KEY` oder `ON MESSAGE` können dem Kernel mitgeteilt werden, so daß eine entprechende Aktion durchgeführt werden kann, wenn eine Aufruftaste betätigt oder eine Nachricht von einem anderen Programm empfangen wird. Das Bridge-Fenster wird vom Kernel direkt kontrolliert und kann, wie bereits erwähnt, vom Programm aus mit einer neuen Titelleiste versehen und zum Anzeigen von interaktiven Benutzermenüs verwendet werden.

Der Bridge-Nachrichtenverwalter ist der zweite Teil der Engine. Er erledigt das Öffnen, Überwachen und Schließen von Mailboxen. Diese Mailboxen werden zum Senden, Empfangen und Lesen von Nachrichten verwendet, die zwischen Anwendungsprogrammen in einem System oder Netzwerk zu Anwendungsprogrammen in einem anderen System gesendet werden.

Der DOS-Supervisor schließlich ist jener Teil des Bridge-Kernels, der die Ausführung von Windows-fremden Anwendungen kontrolliert. Seine Aufgaben beinhalten das Lesen von DOS-Bildschirminhalten, das Abfangen und Abarbeiten von Tastatureingaben und die Verwaltung des Datenaustauschs zwischen Windows-fremden Programmen über die Windows-Zwischenablage.

Der Supervisor erlaubt den Gebrauch von Dialogfeldern während des Ablaufs von normalen DOS-Anwendungen im Full-Screen-Modus. Unter Windows/286 und Bridge/286 ist diese Kontrolle beschränkt auf das Senden einer einzigen Tastaureingabezeile und die Möglichkeit, die Bridge-Engine von DOS-Batchdateien aus mittels `BR.COM` aufzurufen. In *Tabelle 1* werden die verschiedenen Teile und deren Funktion zusammengefaßt.

Es sind APIs (Programmierschnittstellen) für Microsoft Excel, dBASE III und die Programmiersprache C in Entwicklung. Diese APIs werden in der ersten Version noch nicht voll unterstützt. Die Beta-Versionen, die ich sah, bieten Bearbeitungsmöglichkeiten, die denen des Batch-API gleichen und es Microsoft-Excel-Makros ermöglichen, Bridge-Funktionen wie `MSG SEND` oder `MENU LOAD` aufzurufen.

Programmierung mit Bridge

Bridge-Batch stellt, wie jede andere Programmiersprache, verschiedene Kontrollstrukturen zur Verfügung wie `FOR`, `NEXT`, `IF`, und `GOSUB`, sowie das berüchtigte `GOTO`. Die *Tabelle 2* zeigt eine teilweise Liste der Bridge-Befehle.

Variablen brauchen vor Gebrauch nicht definiert zu werden und Bridge/386-Datentypen beinhalten String, Integer, Array, und Fließkomma. Unter den Objekten, derer man sich mit Bridge bedienen kann, sind Menüs, Fenster, Dialogfelder und Mailboxen für die Kommunikation zwischen Anwendungen. Alle normalen Arithmetik-Funktionen sind verfügbar und Vergleiche zwischen Datenobjekten desselben Types können angestellt werden. Es können Variablen in Funktionsaufrufen mittels einer Syntax übergeben werden, die der Schreibweise `%1` der DOS-Batchsprache ähnelt.

Programmier-Befehlsstrukturen

```
call
for
gosub
goto
halt
if
return
```

Batch-Programmierung

```
set
shift
```

Debugging-Unterstützung

```
trace
warning
```

Datei-/DOS-Befehle

```
rename
chdir
delete
rmdir
```

Mailbox-Verwaltung

```
msg create
msg delete
msg netclose
msg netinit
msg query
msg read
msg send
```

Bridge-Fenstersteuerung

```
rows
columns
background
font
vscroll
hscroll
```

Task-Steuerung

```
foreground
background
fullscreen
maximize
minimize
window
```

Tabelle 2: *Batchbefehle von Bridge.*

```
abs
app.exist
array.len
exact
file.size
find
left
upper
```

Tabelle 3: *Funktionen von Bridge.*

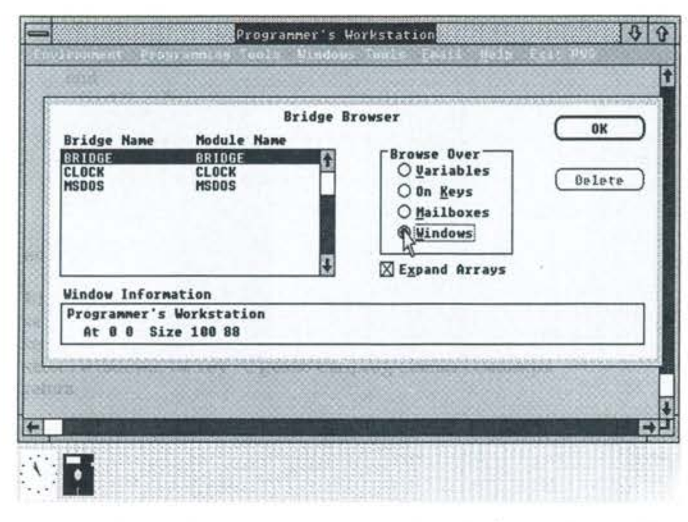


Bild 5: *Der Bridge-Browser zeigt Detailinformationen über das PWS-Programm.*

Mit Bildschirm-Einheiten wird die Position und Größe von Fenstern festgelegt, eine Methode, die mit Prozentanteilen des Bildschirms arbeitet. Individuelle Anwendungen können unter Kontrolle eines Batch-Programms zum Sinnbild verkleinert, wiederhergestellt oder zum Vollbild vergrößert werden.

Es werden eine Menge Funktionen zum Bearbeiten der verschiedenen Datenobjekte zur Verfügung gestellt (siehe *Tabelle 3*, wo eine Teilliste darstellt ist). Ein Beispiel ist die Funktion `FILE_PARSE()`, die zum Auseinandernehmen der Dateinamen-Bestandteile Laufwerk, Pfad, Namen und Erweiterung verwendet wird. Die Array- und Stringoperatoren erinnern an PostScript-Programme, da die Datentypen ähnlich sind.

Viel von Bridges Stärke liegt in den Befehlen zum Bearbeiten von Menüs und Dialogfeldern. Diese Befehle werden zum Aufbau von anpruchsvollen Benutzerschnittstellen und zum Behandeln von Fehlerzuständen, die während der Ausführung von Batchprogrammen entstehen, verwendet. In *Bild 6* ist eine Teiliste der Dialogbefehle aufgeführt.

Die Entwicklung von Bridge

Mehrere Umstände bewegten Softbridge zur Entwicklung von Bridge für die Windows-Umgebung. Der Hauptpunkt waren, laut Ciaramaglia, »Anwendungsprogramme der Weltklasse, wie Microsoft Excel, die für den Windows Presentation Manager entwickelt wurden.« Dies, zusammen mit einem langjährigen guten Verhältnis zwischen Softbridge und Microsoft, welches bis in die Tage von Multiplan im P-Code zurückreicht, war genug Ansporn.

Bridge wurde in C und Assembler mit DOS als Entwicklungsumgebung geschrieben. Ciaramaglia behauptet, daß das wichtigste bei der Entwicklung das gute Verhältnis zwischen Softbridge and Microsoft war. Er sagt weiter, daß »eine Menge DIAL, Zugriff zu den frühen Beta-Versionen

```
DIALOG button_option [default_option] prompt [caption]

    dabei kann button_option sein:

        OK          OKCANCEL        RETRYCANCEL
        YESNO       YESNOCANCEL     ABORTRETRYIGNORE

    und default_option ist:

        DEFBUTTON1      button_number

DIALOG FILE file_spec prompt [caption][RESULT
    variable_name]

DIALOG INPUT prompt [NUMBER][INTEGER][caption][RESULT
    variable_name]

DIALOG LOAD
    BEGIN
        [AT x y]
        [SIZE width height]
        control ...
    END

    dabei kann control sein:

        CAPTION text
        CHECKBOX [AT x y] text [DEFAULT value][RESULT
            variable_name]
        EDIT [WIDTH characters][AT x y][DEFAULT
            text][NUMBER]
            [INTEGER][RESULT variable_name]
        ICON {HAND|EXCLAMATION|QUESTION|ASTERISK}[AT x y]
        PUSHBUTTON [AT x y][DEFAULT value][CANCEL] text [WIDTH
            characters]
        TEXT [AT x y][text][WIDTH characters]
```

Bild 6: *Übersicht der Dialogfeld-Befehle von Bridge.*

und gelegentliche Telephongespräche mit Phil Barrett [dem Entwickler von Windows/386] zur Entwicklung von Bridge /386 notwendig waren«.

Eines der Probleme auf die Softbridge stieß, war die merkwürdige Art des Time-Slicing von Windows beim Multitasking. Die Vordergrund-Task erhält um die 60 Prozent der CPU-Zykluszeit, während die restliche Zeit zwischen den verbleibenden Hintergrund-Tasks aufgeteilt wird. Dadurch wurde es schwierig die Durchführung vorauszusagen, wenn Bridge andere Anwendungen kontrollierte. Eine Anzahl von Erweiterungen wurde auf Ansuchen von Softbridge in Windows eingefügt. Im einzelnen wurden einige der Debug-Funktionen, die planmäßig nach den Beta-Tests entfernt worden wären, beibehalten, da Sie für Bridge/386 so nützlich waren.

Die Entwicklung von PWS

Listing 1 zeigt den vollständigen Quellcode einer einfachen integrierten Programmierumgebung. Die ASCII-Datei `PWSHELP.TXT`, ebenfalls in Listing 1 zu sehen, erscheint in `NOTEPAD`, sobald im PWS-Menü `Help` angewählt wird.

Die Entwicklungszeit dieser Anwendung betrug ungefähr ein Tag und beinhaltete die Zeit zum Lernen von Bridge und das Testen und Debugging von PWS. Ich fand die interaktive Fenstertechnik-Umgebung beim Debuggen

```
REM PWS: Eine Programmierumgebungfür Win/386 und
        Bridge/386
REM Von: Matt Trask Am: 23 APR 88

REM Dieses Programm erwartet folgende Definitionen:
REM Editor, Make-Program, Debugger, und Comm-Program als
REM EDITOR.PIF, MAKE.PIF, DEBUGGER.PIF, und COMMS.PIF.

REM *** Macht das Bridge-Fenster groß genug für die
REM     Menüleiste ***

select bridge; size 100 12; cls

REM *** Lädt das Hauptmenü für die Programmierumgebung ***

caption "Programmer's Workstation"
menu load pws

REM *** bestimmt Hot-Keys ***

select windows; on key ^e gosub callprog editor editor.pif
select windows; on key ^d gosub callprog debugger debugger.pif
select windows; on key ^c gosub callprog comms comms.pif
return


REM *** Hauptmenü der Programmierumgebung ***

menu define pws
begin
    popup En&vironment
     begin
     menuitem "Set Current D&rive"          gosub set_drive
     menuitem "Set Current D&irectory"      gosub set_cwd
     menuitem "Online Communication"        gosub comm
     menuitem "DOS &Command Line"           gosub doscmd
        end

    popup "&Programming Tools"
     begin
     menuitem "    &Edit         "       gosub edit
     menuitem "    &Make         "       gosub make
     menuitem "    &Debug        "       gosub debug
     menuitem "    &SCCS         "       gosub sccs
        end

    popup "&Windows Tools"
     begin
     menuitem &Calculator       gosub callprog calc calc
     menuitem Calen&dar         gosub callprog calendar
                                calendar
     menuitem Cloc&k            gosub callprog clock clock
     menuitem &Notepad          gosub callprog notepad
                                notepad
        end

    popup &Email
     begin
        menuitem "&Check Email"    gosub pwsemail Check
        menuitem "&Send Email"     gosub pwsemail Send
        end

    menuitem &Help              gosub pwshelp

    menuitem "E&xit PWS"        goto pwsexit
end

REM *** Bestimmt das aktuelle Arbeitsverzeichnis ***

:set_cwd
dialog input "Directory name?" result cwd
if %error% == 2 return          ; REM Feld "Abbrechen"
if %cwd% == "" return           ; REM nichts angegeben
cd %cwd%
return
```

von PWS äußerst nützlich. Die Größe der einzelnen Fenster, die von PWS gestartet wurden, konnte so vergrößert oder verkleinert werden, daß der Weg der Programmabarbeitung im Hauptfenster von Bridge verfolgt werden konnte. Wie früher schon erwähnt, kann diese Trace-Option durch die Auswahl von `Programm Trace`, `Message Trace` oder beiden im Bridge-Browser eingeschaltet werden. Der Browser kann jederzeit zur Untersuchung von Bridge-Variablen oder Mailboxen verwendet werden.

Zu den erwähnenswerten Teilen von PWS gehören die drei Zeilen, die die sogenannten »Hot Keys« (Aufruftasten) einführen. Diese Befehle definieren [Ctrl][E], [Ctrl][D], und [Ctrl][C] als Hot Keys, die den Editor, Debugger bzw. ein Kommunikationsprogamm aktivieren.

Der Hauptteil des PWS-Programms befindet sich in den ca. 30 Zeilen nach `MENU DEFINE PWS`. Diese Menüdefinition erzeugt die veschiedenen Pull-Down-Menüs und andere Auswahlmöglichkeiten des Menüs, die die Programmierumgebung steuern. Nach dem Start von PWS wird der Programmierer gewöhnlich `Set Current Drive` und `Set Current Directory` anwählen, um Standardwerte für die Edit-, Make-, und Debug-Befehle einzustellen. `Edit` und `Debug` zeigen den Gebrauch von Dialogfeldern bei der Auswahl einer Datei im aktuellen Verzeichnis.

PWS ist ein ziemlich einfaches Beispiel für die Batch-Fähigkeiten von Bridge/386. Es kann durch den Gebrauch von BRDOS (dem DOS-Supervisor), anstelle der PIF-Dateien zum Starten der Programme Edit, Make, und Debug erweitert werden, wodurch größere Möglichkeiten zur Kontrolle und Kommunikation zwischen Programmen geboten würden. Es könnte auch der Programmcode so geschrieben werden, daß die Existenz von wichtigen Dateien, wie beispielsweise `EDITOR.PIF`, geprüft und automatisch PIFEDIT gestartet wird, wenn dies nicht der Fall ist. Mit einer solchen automatischen Einrichtung kann jedes Bridge-Programm ausgestattet werden.

Fazit

Bridge/286 und Bridge/386 sind wahrscheinlich am besten für System-Integrierer geeignet, die Komplettanwendungen entwickeln, die Windows/286 oder Windows/386 beinhalten. Größere Betriebe könnten Bridge/286 und Bridge/386 zur Konfiguration individueller PCs mit vereinfachten Benutzerschnittstellen, die für die Angestellten leichter verständlich sind, verwenden. Softwareanbieter könnten Bridge zur automatischen Installation und als Online-Tutorial für ihre eigenen Produkte gebrauchen.

PWS sollte als schnelles und leichtes Beispiel der Bridge/386-Programmierung verstanden werden, und nicht als Darstellung aller Bridge-Fähigkeiten. Ich könnte mir unter Bridge/386 eine Arbeitsumgebung vorstellen, die ihre Aufgaben daran anpaßt, ob ich gerade als Programmierer, Autor oder Buchhalter tätig bin.

```
REM ***  Bestimmt das aktuelle Laufwerk ***

:set_drive
dialog input "New drive name?" result drive
if %error% == 2 return          ; REM Feld "Abbrechen"
if %drive% == "" return         ; REM nichts angegeben
%drive%:
return


REM ***  Startet COMMAND.COM  ***

:doscmd
exec /N:doscmd commandw.pif
return


REM ***  Startet eine Kommunikations-Sitzung  ***

:comm
exec /N:comms comms.pif
return


REM ***  Editiert eine Datei ***

:edit
dialog file *.asm "Edit which file?" result edfile
if %error% == 2 return          ; REM Feld "Abbrechen"
exec editor.pif %edfile%
return


REM ***  Erstellt ein Programm ***

:make
exec make.pif
return


REM ***  Debugging eines Programms ***

:debug
dialog file *.exe "Debug which program?" result dbgfile
if %error% == 2 return          ; REM Feld "Abbrechen"
exec debugger.pif %dbgfile%
return

REM ***  Startet das Quellcode-Kontrollsystem. ***

:sccs
dialog ok "Not implemented yet" "SCCS"
return

REM ***  Unterprogramm zum Aufruf einer Windows-Anwendung ***

:callprog
REM Sollte die Anwendung bereits existieren, wird sie aktiviert
REM und zurückgekehrt,
if app.exist(%1) select %1; activate; return
REM ansonsten wird sie gestartet
exec /N:%1 %2
return

REM ***  Ausgabezeile für Email-System ***

:pwsemail
dialog ok "Email system not available in PWS demo" "%1 Email"
return

REM ***  Zeigt Hilfstext mittels Notepad ***

:pwshelp
menu load    ; REM Lädt Beendigungs-Hilfemenü in das
             ; Bridge-Fenster
begin
    menuitem "E&xit Help" menu load pws; warning off;
    select help; close
end
exec /N:help notepad.exe pwshelp; move 0 12 100 88
return
```

```
REM *** Schließen der Anwendung und Ende ***

:pwsexit

if app.exist(comms) then
    select comms; activate
    dialog ok "Exit halted, you must close this app
    first" Comms
    halt
end if

if app.exist(editor) then
    select editor; activate
    dialog ok "Exit halted, you must close this app
    first" Comms
    halt
end if

if app.exist(debugger) then
    select debugger; activate
    dialog ok "Exit halted, you must close this app
    first" Comms
    halt
end if

REM *** Freigabe der Hot-Keys ***

warning off; select windows; on key ^c
warning off; select windows; on key ^d
warning off; select windows; on key ^e

REM *** Schließen der Windows-Anwendungen ***

warning off; select calc; close
warning off; select calendar; close
warning off; select clock; close
warning off; select notepad; close

REM *** Windows beenden ***
warning off; select msdos; close

.
```

```
    Help Text for PWS, the Programmer's Workstation


This is an example of a programmer's workstation including:

    ·   Setting the default drive and working directory

    ·   Running a communications program as a background
        task

    ·   Running a DOS shell and command line

    ·   Some desktop tools

    ·   A link to an electronic mail system

    ·   Help documentation specific to the workstation

    ·   "Hot keys" for quickly switching between
        applications

COMMANDS:

    Set Current Drive - use this to log the drive
            containing your project's source tree.

    Set Current Directory - specify the current working
            directory for your project.

    Online Communication - start a comm program that can
            be used for background file
            transfers while working.

    DOS Command Line - runs COMMAND.COM in a virtual
            machine for normal access to DOS
            capabilities.

    Programming Tools - edit, make, debug, and SCCS the
            parts of your current project. These
            commands assume that you have already
            set the current drive and directory.
            You must create PIF files for each of
            these choices that define your
            particular tools.

    Windows Tools - Provides access to four desktop
            tools-calculator, calendar, clock, and
            notepad.

    Email - Allows checking or sending of electronic
            mail messages. (The email system is not
            available for this demo.)

    Help - Displays this help information.

    Exit - Closes all open applications and exits from
            PWS.

HOT KEYS:

    The following hot keys are available to quickly
    switch to a specific application.

     Ctrl-C (for Comms)     - switch to the communication
                              session
     Ctrl-D (for Debugger)  - switch to the current
                              debugging session
     Ctrl-E (for Editor)    - switch to the current edit
                              session
```

Listing 1: `PWS.BR`, *eine integrierte Programmierumgebung.*

Eine faszinierende Möglichkeit wäre die Entwicklung eines automatischen Aktiensystems, das die verschiedenen Notierungen von einem Online-Börsentelegraph einliest, und nachdem die Daten in einem Microsoft-Excel-Arbeitsblatt behandelt wurden, Kauf- oder Verkaufsentscheidungen trifft, und BRDOS zur Steuerung eines Kommunikationsprogramm verwendet, das die Aufträge an den Computer eines Börsenmaklers sendet.

Obgleich ich noch nicht herausgefunden habe, wie ich zwei Vorkommen von Reversi gegeneinander spielen lassen kann, scheint es doch genug Möglichkeiten in Bridge zu geben, alles andere auszuführen, was ich mir unter Windows wünsche. Softbridge Microsystems hat mit Bridge Windows etwas zu Windows hinzugefügt, das für viele Benutzer von Personalcomputern interessant ist. Bridge /386 und Bridge/286 werden eventuell die Art verändern, in der Leute Microsoft Windows/386 und Windows/286 verwenden.

Matt Trask

Mitteilungen .... Mitteilungen .... Mitteilungen

Philips, Sony und Microsoft entwickeln erweiterte CD-ROM-Architektur

Die Unternehmen Philips, Sony und Microsoft haben eine Vereinbarung über die gemeinsame Entwicklung eines erweiterten CD-ROM-Formats mit der Bezeichnung »CD-ROM Extended Architecture« (CD-ROM-XA) getroffen. Das erweiterte Format umfaßt die Audio- und Grafiktechnik des CD-I-Formats und dient als Brücke zwischen CD-ROM und CD-Interactive (CD-I).

Das CD-ROM-Format ist von Philips und Sony ursprünglich als offenes System für die Datenspeicherung entwickelt worden. Als neue Aufgaben und Anforderungen entstanden, entwickelte eine Gruppe von Unternehmen der Computerindustrie eine gemeinsame Speicher- und Dateistruktur, die mittlerweile unter der Bezeichnung ISO9660 ein internationaler Standard geworden ist.

In jüngster Zeit gibt es einen dringenden Bedarf für die Definition weiterer CD-ROM-Format-Standards, die mit dem ISO9660-Standard übereinstimmen. Dies betrifft besonders den Bereich der Multimedia-Applikationen für PCs. Betroffen sind nicht nur Texte und Daten, sondern auch die komprimierte Speicherung von Tönen, Grafiken, stehenden Bildern und möglicherweise auch bewegten Bildern.

Die drei Unternehmen Philips, Sony und Microsoft haben sich auf folgendes Entwicklungskonzept für CD-ROM-XA verständigt:

- »Interleaved ADPCM-Audio« - wie im CD-I-Standard definiert - wird auch Bestandteil des CD-ROM-XA sein. Detaillierte Spezifikationen für diesen Teil des Standards werden in Kürze veröffentlicht.

- Im Rahmen von CD-ROM-XA wird zusätzlich ein Text/Grafik-Darstellungsformat für PC-Bildschirme definiert. Ein erstes Vorschlags-Referenzdokument für das vollständige Format soll noch in diesem Jahr verbreitet werden. Die endgültigen Spezifikationen sollen im nächsten Jahr folgen.

CD-ROM-XA bietet eine Vielzahl von Funktionen, wie sie auch im CD-I-Format zu finden sind. Diese Funktionen sind allerdings nicht abhängig von einem speziellen Betriebssystem oder einer bestimmten CPU. Das neue Format wird Programmanbietern die Möglichkeit geben, Disks auf den Markt zu bringen, die nicht nur auf jedem entsprechend ausgestatteten PC, sondern auch auf jedem CD-I-System abspielbar sind. Diese Möglichkeit wird dazu beitragen, die Voraussetzungen dafür zu schaffen, daß das CD-I-System in naher Zukunft unter günstigen Bedingungen eingeführt werden kann.

Das CD-Digital-Audio-Format soll auch weiterhin konventionellen Musikapplikationen vorbehalten bleiben, um mögliche Konflikte mit der Musikbranche zu vermeiden.

CD-ROM-XA wird sowohl den privaten, wie den professionellen Anwendern eine neue Dimension von CD-Applikationen erschließen. Als Brücke zwischen CD-ROM und CD-I erlaubt das neue Format den Firmen Philips und Sony, die derzeitigen Marktchancen des CD-ROM-Geschäftes zu fördern, während gleichzeitig das Feld für die Einführung des CD-I-Systems bereitet wird.

Fakten zum CD-I-Standard

Als gemeinsame Entwicklung von Philips und Sony unterstützt CD-I auf breiter Basis die interaktive Nutzung und die Anwendung von Daten und Informationen in Form von Tönen (Musik, Sprache), Video-Standbildern, Zeichentrickfilmen, Grafiken und Computerprogrammen. Wie CD-Audio basiert auch CD-I auf internationalen Standards, die eine vollständige Kompatibilität sichern, und ermöglicht eine Vielzahl von Anwendungen im privaten Bereich, im Bildungswesen und auf anderen Gebieten.

Genau wie ein konventioneller CD-Audio-Player kann ein CD-I-Player auf einfache Weise an eine Stereo-Anlage, oder - wie ein Videorekorder - an ein Fernsehgerät angeschlossen werden. Ein CD-I-Player ist außerdem einfach zu bedienen. Er ist mit Mikroprozessoren der 68000-Serie ausgestattet und arbeitet mit dem Betriebssystem CD-RTOS (CD Real-Time OS). Zur interaktiven Nutzung einer breiten Palette von Software haben CD-I-Player einen eingebauten Systemcontroller. Der Systemcontroller schafft die Voraussetzung zur Entwicklung und Anwendung eines breiten Spektrums von Software für die Unterhaltung, die Information oder zur Ausbildung.

Fakten zum CD-ROM-XA-Standard

CD-ROM-XA wird mit der ADPCM (Adaptive Differential Pulse Code Modulation) arbeiten, so wie sie beim CD-I-System standardisiert ist. Diese Modulationsart unterstützt die komprimierte Speicherung von Digital-Audio-Signalen in Interleaved-Technik. Texte, Grafiken und Video-Standbilder lassen sich auf diese Weise miteinander wiedergeben, so daß Echtzeit-Simultan-Playback-Aufnahmen möglich sind.

Das CD-Digital-Audio-Format (Red Book) wird auch weiterhin konventionellen Musikanwendungen vorbehalten sein, so daß jeglicher Konflikt mit der Musikbranche vermieden wird.

Zusätzlich zur Interleaved-ADPCM-Audio-Funktion wird das CD-ROM-XA-Format zukünftig einen Text/Grafik-Bildschirm-Darstellungs-Standard beinhalten. Das System zielt hauptsächlich auf professionelle und institutionelle Anwender sowie auf Referenzanwendungen. Es ist grundsätzlich transparent für jeden PC und nicht abhängig von dem Einsatz spezieller Mikroprozessoren oder Betriebssysteme.

Programmanbieter werden dazu ermuntert, Disks gemäß dem neuen Format zu erstellen, die mindestens auf MS-DOS- und CD-RTOS/68K-Systemen abspielbar sind.

Mitteilungen .... Mitteilungen .... Mitteilungen

Die wesentlichen Systemanforderungen umfassen:

- Jede CD-ROM-XA-Spur ist eine CD-ROM-Mode-2-Spur.
- ADPCM gemäß CD-I-Spezifikationen mit Tonqualitätsstufen B und C.
- Bildschirm-Darstellungsformat: Auflösung 640 x 480 Bildpunkte, maximal 8 Bit Tiefe
- Farbcodierung CLUT, maximal 8 Bit.
- Speicher- und Dateistruktur gemäß ISO9660.
- Zeichensatz gemäß ISO8859/1.
- »Single Plane«-Darstellung, Mischung von Text und Grafiken.
- Für jedes Zielsystem Retrieval-Software auf der Disk.

Software-Entwicklungs-Kit für den SQL-Server

Microsoft und Ashton-Tate haben jetzt ein Software-Entwicklungs-Kit für den Ashton-Tate/Microsoft-SQL-Server vorgestellt. Der SQL-Server, eine Gemeinschaftsentwicklung der beiden Unternehmen, ist eine auf MS OS/2-basierende Software-Plattform, die Netzwerk-Anwendern hochentwickelte Datenbank-Struktur- und Zugriffshilfen zur Verfügung stellt. Das Entwicklungspaket mit der Bezeichnung »SQL Server Network Developer's Kit« enthält die komplette Software, Dokumentation und die »Application Programming Interface Libraries« für den SQL-Server und den MS OS/2-LAN-Manager, das neue Betriebssystem für lokale Netzwerke von Microsoft. Die API-Libraries können zusammen mit dem C-Compiler von Microsoft eingesetzt werden, um Applikationen für MS-DOS, Microsoft Windows oder MS OS/2 zu schreiben, die mit dem SQL-Server in einem Netzwerk arbeiten.

»Das 'SQL Server Network Developer's Kit' versetzt die Software-Entwickler in die Lage, aus einer günstigen Startposition heraus Applikationen für den Microsoft-SQL-Server zu entwickeln, der gegen Ende dieses Jahres lieferbar sein wird. Wir meinen, daß die Software-Entwickler in dem SQL-Server und dem MS OS/2-LAN-Manager die Lösung für komplexe Aufgaben bei Multiuser-Datenbanken in lokalen Netzwerken sehen. Wir freuen uns, daß unsere strategische Allianz mit Ashton-Tate dazu führt, diese Technik den Entwicklern frühzeitig zur Verfügung zu stellen«, so Christian Wedell, Geschäftsführer der deutschen Microsoft GmbH, zur Ankündigung des SQL-Server-Entwicklungs-Paketes.

Der SQL-Server erlaubt in Verbindung mit dem OS/2-LAN-Manager den Aufbau komplexer Anwendungsprogramme, einschließlich verteilter PC-Datenbanken, Tabellenkalkulationen und Buchhaltungssystemen. Der SQL-Server wird auf einem Server innerhalb des lokalen Netzwerkes installiert und setzt als Betriebssystem-Umgebung auf dem Server MS OS/2 und ein MS OS/2-Netzwerk-Betriebssystem, wie den MS OS/2-LAN-Manager, voraus.

Programmiersprachen-Updates auf CD-ROM

Microsoft hat kürzlich angekündigt, daß zukünftige Updates von ausgewählten Microsoft-Programmiersprachen alternativ zu Floppy-Disks und gedruckten Dokumentationen auch auf CD-ROM-Disks erhältlich sein werden. Außerdem wurde die neue Microsoft-»Programmer's Library« vorgestellt, ein leistungsfähiges Hilfsmittel, das Programmierern Online-Zugriff auf eine umfassende Auswahl von Büchern, technischen Handbüchern und Programmen verschafft, die auf einer einzigen CD-ROM-Disk gespeichert sind.

»Wir sind sehr angetan von den Möglichkeiten der CD-ROM als einem Medium, das es erlaubt, große Mengen von Texten sowie auch Compiler und Entwicklungs-Umgebungs-Software zu vertreiben und zu verwalten«, so Christian Wedell, Geschäftsführer der deutschen Microsoft GmbH, zur Ankündigung der Microsoft-»Programmer's Library«.

CD-ROM erschließt für den Anwender ein völlig neues Speichermedium. Er hat nun die Möglichkeit, die neuesten Microsoft-Compiler, Entwicklungs-Werkzeuge und Dokumentationen auf einer einzigen Disk bereit zu haben, während die bisherige Form der Speicherung auf Floppy-Disks rund 20 Disketten plus der gedruckten Dokumentation erfordert. Sobald der Anwender die neue CD-ROM erhalten hat, kann er den Inhalt auf eine Festplatte umkopieren und in der üblichen Weise damit arbeiten. Die Kapazität einer CD-ROM-Diskette von 660 Mbyte, im Vergleich zu 1,2 Mbyte bei einer herkömmlichen Floppy-Disk, erlaubt es Microsoft, CD-ROM-Anwendern einen zusätzlichen Vorteil zu bieten: Programm-Installationen und -Konfigurationen lassen sich vereinfachen. Microsoft nutzt dafür die zusätzlich zur Verfügung stehende Kapazität, um auf der CD-ROM konfigurierte Versionen der Programme für unterschiedliche Speichergrößen und Hardware-Konfigurationen zu liefern.

Die Programmer's Library auf CD-ROM

Mit der Microsoft-»Programmer's Library« steht den Entwicklern von PC-Software nun ein leistungsfähiges Werkzeug zur Verfügung, das auf einer CD-ROM mehr als 20.000 Seiten Dokumentation und rund 1.200 Musterprogramme bereitstellt. Die 48 Handbücher sowie andere technische Dokumentationen der »Programmer's Library« - gespeichert auf einer CD-ROM-Disk - beinhalten Informationen über die Betriebssysteme und Programmiersprachen von Microsoft. Das Spektrum reicht dabei von relativ kurzen Hinweisen bis zu detaillierten Abhandlungen. Für die gesamte Text-Datenbank gibt es ein Inhaltsverzeichnis und Querverweise, so daß das Auffinden und Verbinden von Informationen relativ einfach ist. Die Texte lassen sich mit Hilfe eines Texteditors oder eines Textverarbeitungspro-

Mitteilungen . . . . Mitteilungen . . . . Mitteilungen

gramms vom Anwender aufrufen und direkt in Programme oder Dokumente hineinkopieren. Direkt werden die neuen gängigsten Programm-Editoren sowie fünf Textverarbeitungsprogramme unterstützt. Auch die Übernahme in Form von ASCII-Files ist möglich.

Die CD-ROM-Disk enthält darüber hinaus eine Reihe von Informationen über Hardware, wie die Microsoft-Maus, CD-ROM-Laufwerke und Videokarten. Die »Programmer's Library« läßt sich sowohl direkt als auch aus anderen Programmen aufrufen. Im Gegensatz zu speicherresidenten Programmen kann der Anwender die »Programmer's Library«-Texte jedoch auf einfache Weise wieder aus dem Speicher löschen, um Platz für andere Programme oder Daten zu schaffen. Einen Nutzen aus der »Programmer's Library« können nicht nur Programmierer ziehen, sondern alle, die mit Software arbeiten, z.B. Autoren technischer Beiträge oder Produkt-Support-Spezialisten, die schnell auf Informationen zugreifen wollen, ohne umfangreiche Handbücher durchblättern zu müssen.

Die Bestandteile der »Programmer's Library«

Das Inhaltsverzeichnis der »Programmer's Library« ist in neun Kapitel gegliedert: MS OS/2, Microsoft Windows, MS-DOS, Microsoft C, Microsoft BASIC, Microsoft Makro-Assembler, Microsoft PASCAL, Microsoft Fortran und Hardware. Jeder Bereich umfaßt alle technischen Handbücher der jeweiligen Microsoft-Version des Betriebssystems oder der Programmiersprache.

Die Dokumentation für den Presentation Manager und den LAN-Manager war zur Zeit der Fertigstellung der ersten Ausgabe noch nicht verfügbar, wird aber Anfang 1989 in der nächsten »Programmers' Library«-Ausgabe enthalten sein. Des weiteren enthält die »Programmer's Library« 2.700 Dateien mit Musterprogrammen, die ebenfalls in einem Inhaltsverzeichnis aufgelistet sind und von hier aus gesucht, kopiert und an anderer Stelle eingefügt werden können.

Die Suche in der »Programmer's Library«

Kernpunkt der Microsoft-»Programmer's Library« ist eine hochentwickelte Suchfunktion. Der Anwender gibt nur verschiedene Suchbegriffe ein und spezifiziert die logischen Beziehungen zwischen ihnen. Dabei läßt sich vorgeben, ob eine Ausgabe erfolgen soll, wenn einige der Begriffe gefunden wurden, oder ob nur dann eine Ausgabe erfolgen soll, wenn alle Begriffe gefunden wurden, oder daß nur dann eine Ausgabe erfolgt, wenn die Suchfunktion auf eine exakt definierte Zeichenfolge gestoßen ist. Der Anwender kann aber die Suchfunktion auch so einsetzen, daß verschiedene Begriffe im selben Kapitel oder im selben Absatz zu finden sind.

Wenn die »Programmer's Library« aus einem Texteditor oder einem Textverarbeitungsprogramm heraus angesprochen wird, so erfolgt die Suche automatisch nach dem Begriff, auf dem der Cursor gerade steht. Der Anwender braucht dazu lediglich die Eingabetaste zu drücken, um die Suchfunktion einzuschalten. Er hat wie bei einem Buch die Möglichkeit, im Inhaltsverzeichnis ebenso wie in den aufgerufenen Texten zu »blättern«, um sich einen Überblick zu verschaffen. Damit der Suchaufwand und die Suchzeit in vernünftigen Grenzen bleiben, erfolgt die Suche immer in einem Teilbereich des gesamten Inhalts. Der Anwender hat jedoch die Möglichkeit, schnell zwischen den Teil-Inhaltsverzeichnissen und den Bereichen hin- und herzuschalten, indem er die dafür zur Verfügung stehenden Such- und Bibliothekskommandos benutzt.

Kompatibilität zu zahlreichen Editoren und Textverarbeitungs-Programmen

Die Kompatibilität der »Programmer's Library« zu einer Vielzahl von Texteditoren und Textverarbeitungsprogrammen macht es möglich, daß Teile aus der Bibliothek mit Hilfe der »Copy and Paste«-Funktion in das Anwendungsprogramm oder das jeweils bearbeitete Dokument übernommen werden. Kompatible Texteditoren sind BRIEF 2.0, Epsilon-Editor 3.21, Microsoft Editor, Microsoft QuickBASIC-Editor 4.0, Microsoft QuickC 1.0, »The Norton«-Editor 1.3C, Turbo-Pascal-Editor 4.0, Turbo-C 1.5 und VEDIT-Plus 2.03F. Kompatible Textverarbeitungsprogramme sind Microsoft Word 2.0 oder höher, PC-Write 2.71, WordPerfect 4.2 und 5.0, WordStar 4.0 und XyWrite III und III+.

Weitere spezielle Merkmale und Funktionen

Quick Reference: Kurze Definitionen und Erklärungen, verfügbar mit Hilfe des »Quick Reference«-Menü-Optionskommandos. QuickRef führt den Anwender direkt zur Hauptdefinition oder Erklärung eines Begriffs und vermeidet dabei die Suche in einer Liste von Varianten des Begriffs. Die »Quick Reference«-Funktion steht neben den Online-Hilfepaketen zur Verfügung, die für alle neun Datenbank-Bereiche vorhanden sind.

Lesezeichen (Bookmarks): Elektronische »Lesezeichen« ermöglichen es dem Anwender, direkt eine gekennzeichnete Textpassage aufzurufen, ohne dabei das ganze Inhaltsverzeichnis durchblättern zu müssen. Der Anwender hat außerdem die Möglichkeit, zu jedem Lesezeichen einen Hinweis oder einen Kommentar zu speichern, der hilft, sich an die entsprechende Textpassage zu erinnern und es auch anderen Benutzern vereinfacht, solche Textpassagen aufzurufen. Bis zu 80 »Lesezeichen« per Workstation sind möglich.

Querverweise (Cross References): Eine wichtige Funktion innerhalb der »Programmer's Library« ist die Möglichkeit, auf einfache Weise von einer Textpassage per Querverweis in eine andere Textpassage zu springen. Der Anwender kann über eine Kette von bis zu 20 Querverweisen in Texten blättern, wieder zurückzublättern oder mit einem einzigen Tastendruck in den Ursprungstext zurückgehen.

Mitteilungen .... Mitteilungen .... Mitteilungen

Lokalisierung (Location): Die Lokalisierungsfunktion gibt dem Anwender mit einem einzigen Tastendruck Auskunft darüber, in welchem Teil der Library er sich aktuell befindet. Ist dieser Teil ein Dokument, so gibt die Lokalisierungsfunktion den Teil, Bereich, Unterbereich und den Absatz an. Sofern der Anwender sich in einem Beispielprogramm befindet, spezifiziert die Lokalisierungsfunktion das Directory, die Datei und die Funktion.
EMS-Unterstützung: Die »Programmer's Library« unterstützt die erweiterte Speicherspezifikation (EMS = Expanded Memory Specification).
Netzwerk-Unterstützung: Mehrere Anwender haben über einen einzigen CD-ROM-Server Zugriff auf die »Programmer's Library«. Es stehen zwei Netzwerk-Lösungen für CD-ROM zur Verfügung: OPTI-NET - das mit dem NetBIOS-Netzwerk arbeitet - von der Firma Online Systems sowie CD-NET, eine Hardware-/Software-Lösung der Firma Meridian Data, Santa Cruz, Kalifornien.
Handbücher: Die Microsoft-»Programmer's Library« wird mit einer Bedienungsanleitung geliefert, die sowohl in gedruckter Form als auch Online-computergestützt vorhanden ist.
Hilfefunktion: Für die schnelle Hilfe bei der Arbeit mit der »Programmer's Library« erscheint nach Drücken einer entsprechenden Taste eine Erklärung des jeweils aktuellen Menüpunkts. Detailliertere Hilfsinformationen erhält der Anwender durch die Bedienungsanleitung, in der er genauso suchen und blättern kann, wie im Inhaltsverzeichnis der »Programmer's Library«.

Als Hard-/Software-Ausstattung setzt die »Programmer's Library« IBM-PC, PC/XT, PC/AT oder Personal-System/2-Computer bzw. 100%-kompatible Systeme mit 640-Kbyte-Hauptspeicher und MS-DOS-3.1, 3.2- oder 3.3-Betriebssystem voraus. Um die Zugriffszeit erheblich zu verkürzen, empfiehlt Microsoft den Einsatz einer Festplatte. Da die »Programmer's Library« mit virtueller Speicherverwaltung arbeitet, hängt die Performance weitgehend von Disketten-/Festplatten-Zugriff ab. Die »Programmer's Library« läßt sich allerdings auch auf einem PC nutzen, der nur über zwei doppelseitige Floppy-Disk-Laufwerke verfügt. Als weitere Hardware ist ein CD-ROM-Disk-Laufwerk mit MS-DOS-CD-ROM-Extension erforderlich. Zukünftige Versionen der »Programmer's Library« werden auch unter MS OS/2 arbeiten.

Die neue »Extended Memory Specification 2.0«

Die Firmen Lotus, Intel-PCEO, Microsoft und AST Research gaben in diesen Tagen die »Extended Memory Specification 2.0« (XMS 2.0) bekannt. Die neue Version wurde von den vier Unternehmen gemeinsam entwickelt und ist die erste Spezifikation, auf die sich diese Unternehmen gemeinsam offiziell festgelegt haben. Die XMS-Spezifikation wird nach Meinung von Christian Wedell, Geschäftsführer der deutschen Microsoft GmbH, sowohl der Computerindustrie wie den PC-Anwendern großen Nutzen bringen. Microsoft hat - basierend auf der XMS-Spezifikation - wesentliche Verbesserungen bei der grafischen Bedienungsoberfläche Microsoft Windows durchgeführt und geht davon aus, auch bei zukünftigen Produkten Vorteile aus der XMS-Spezifikation ziehen zu können.

Bei der Firma Lotus ist Frank Ingari, Vice President Marketing, der Meinung, daß Anwender von Tabellenkalkulations-Programmen mehr als alle anderen Benutzer von Applikations-Programmen einen Gewinn durch den effizienten und flexiblen Einsatz eines großen Arbeitsspeichers haben werden. »Die neue Spezifikation hilft Lotus, besonders den Anforderungen derjenigen Anwender entgegenzukommen, die Anwendungen bearbeiten, wo es auf die Konsolidierung und Analyse von Informationen aus vielen Quellen ankommt«, so Ingari.

Die gemeinsame Entwicklung des XMS 2.0 durch vier führende Unternehmen der PC-Branche ist nach Ansicht von Jim Johnson, General Manager der Intel-PCEO, auch ein unmittelbarer Vorteil für Entwickler, weil hier ein Industriestandard für den Einsatz von erweiterten Hauptspeichern in PCs auf 80286- und 80386-Prozessorbasis gesetzt wird. Die Firma Intel will in diesem Zusammenhang sicherstellen, daß Programme, die für die Nutzung der EMS- und XMS-Spezifikationen ausgelegt sind, auch auf den Intel-»Above Boards« laufen.

Die XMS-Version 2.0 bietet den Programmierern ein Applikations-Interface, das einen erweiterten Hauptspeicher oberhalb der 1-Mbyte-Adressierung von 80286- und 80386-Computern möglich macht. Ein XMS-2.0-Treiber sorgt dabei für die Verfügbarkeit und Verwaltung des erweiterten Speichers bei Applikationsprogrammen, die mit dieser Speichererweiterung arbeiten. Der Anwender profitiert darüber hinaus noch in einer anderen Weise von der XMS-Spezifikation. Sie bietet nämlich einen zusätzlichen 64-Kbyte-Speicher oberhalb der 640-Kbyte-Grenze. Diese Funktion verbessert bei Computern auf 80286- und 80386-Prozessorbasis die Geschwindigkeit und Verfügbarkeit von MS-DOS-Programmen. Microsoft-Windows 2.1 ist eines der ersten Programme, bei dem die Möglichkeiten des XMS-Standards genutzt werden.

Nach Darstellung von Duane Cowgill, Marketing-Product-Manager der Firma AST Research, erlaubt es XMS 2.0, daß Applikationsprogramme unterschiedlicher Software-Entwickler in XMS- und EMS-Speichern koexistieren. Wegen der erweiterten Speichermöglichkeiten von PCs auf 80286- und 80386-Basis, des 64-Kbyte-Zusatzspeichers für DOS und des Industriestandards, der durch die XMS-Version 2.0 geschaffen wurde, wird AST Research die XMS-Version 2.0 auf allen seinen »Premium Computern« implementieren.

Neben den bereits erwähnten Unternehmen unterstützen eine Reihe weiterer Anbieter von Hard- und Software

Mitteilungen .... Mitteilungen .... Mitteilungen

aus dem PC-Bereich den neuen XMS-2.0-Standard. Zu diesen Firmen gehören unter anderem Qualitas, mit dem Produkt 386-to-the-Max, Phar Lap Software, mit dem Produkt 386/DOS-Extender, und andere.

Microsoft nimmt Stellung zu EISA

Die Verfügbarkeit der EISA-Spezifikation (Erweiterte Industriestandard-Architektur) und deren Billigung durch die Hardware-Hersteller ist ein bedeutendes Ereignis für Microsoft. Die hinter der EISA-Spezifikation stehenden Unternehmen sind alle OEM-Kunden von Microsoft. Daher stellt Microsoft selbstverständlich System-Software-Produkte für seine EISA-Partner und deren neuen Standard bereit. Dies ist auf einfache Weise möglich, da die bereits existierenden Microsoft-System-Software-Produkte ohne großen Aufwand an Computer nach EISA-Standard anpaßbar sind.

In Zukunft werden die System-Software-Produkte von Microsoft für MS-DOS - mit und ohne Microsoft Windows - für MS OS/2 und für XENIX/UNIX so ausgelegt sein, daß sie die neuen Hochgeschwindigkeits-32-Bit-Einheiten, wie zum Beispiel Hochleistungs-Disk-Subsysteme und Netzwerk-Interfaces unterstützen. Microsoft wird deshalb die Möglichkeiten voll nutzen, die sich durch die 32-Bit-Einschubkarten nach EISA-Spezifikationen bieten.

Die Möglichkeit, daß Computer auf EISA-Basis mit mehreren Prozessoren arbeiten können, ist ebenfalls sehr wichtig. Auch diese Fähigkeit wird durch die System-Software von Microsoft ausgenutzt. Die erste System-Software, bei der die Multiprozessor-Funktion zum Tragen kommt, ist der Microsoft-OS/2-LAN-Manager, bei dem vor allen Dingen die Leistungsvorteile eines Multiprozessor-Systems in einer Server-Konfiguration für lokale Netzwerke genutzt werden. Sowohl Microsoft als auch die Computer-Hersteller betrachten Computer auf EISA-Basis als günstige Möglichkeit, Software zu bieten, welche den Anwender bei der Arbeit mit dem Computer unterstützt. Dies gilt zum Beispiel für die Integration von Standard-Setup- und Konfigurationshilfen in die Betriebssysteme. Von einer solchen Möglichkeit profitieren sowohl die Hardware-Hersteller wie auch die Endanwender.

MACH-20 wird ausgeliefert

Microsoft liefert ab November 1988 eine spezielle Adaption des Multitasking-Betriebssystems MS OS/2 aus. Das MACH-20-System ist eine Einschubkarte, die in IBM-PCs (oder Kompatiblen) nur einen einzigen Kartensteckplatz belegt und deren Leistung, Speicherverfügbarkeit und Einsatzmöglichkeiten erheblich verbessert.

Die Basiseinheit des MACH-20-Systems ist das »Performance Enhancement Board«, auf dem sich ein Intel-80286-Prozessor mit 8 MHz Taktfrequenz, ein 16-Kbyte-Cache-Speicher und ein 16-Bit-Datenbus befinden. Darüber hinaus hat das Board einen »InPort«-Anschluß, der es ermöglicht, eine Maus anzuschließen, ohne daß dafür ein weiterer Einsteckplatz oder Port belegt wird.

Auf die Basiskarte ist die »Memory Plus Option« steckbar, die einen zusätzlichen Speicher und erweiterte Speicheradressierfähigkeiten bietet. »Memory Plus« unterstützt sowohl den neuesten Speicherstandard LIM 4.0 wie auch frühere Versionen dieses Standards. Sie ist mit 512 Kbyte RAM bestückt und kann bis zu 3,5 Mbyte aufnehmen.

Eine weitere Einsteck-Einheit für das Basisboard ist die »Disk Plus Option«; ein Multimedia-Controller zur Steuerung von zwei internen Floppy-Disk-Laufwerken. »Disk Plus« erlaubt die Installation eines MACH-20-Systems in den Slot des bisherigen Floppy-Disk-Controllers (falls kein anderer Einsteckplatz verfügbar ist). Darüber hinaus kann der Anwender dank der »Disk Plus Option« 5,25- sowie 3,5-Zoll-Laufwerke mit hoher Speicherkapazität verwenden und ist mit seinen Disketten kompatibel zu den neuesten PCs.

Da es in der Architektur zwischen dem IBM-PC/AT und der vorherigen PC-Generation wesentliche Unterschiede gibt, läuft das Betriebssystem MS OS/2 nicht auf PC- und XT-Maschinen. Die Adaption des MS OS/2, zusammen mit dem MACH-20-Performance-Enhancement-Board und der »Memory Plus Option« versetzt den Anwender in die Lage, mit MS OS/2 auf seinen bisherigen PC- oder XT-Computern zu arbeiten.

Christian Wedell, Geschäftsführer der deutschen Microsoft GmbH: »Wir wissen, daß viele PC-Anwender einen längerfristigen Nutzen aus ihren vorhandenen Computern ziehen möchten. Das MACH-20-System in Kombination mit der MS OS/2-Adaption verschafft ihnen dazu die Möglichkeit. Den Anwendern steht nun eine Lösung zur Verfügung, die eine gleich hohe oder sogar bessere Leistung bietet als ein IBM-PC/AT. Und: Das MACH-20-System erlaubt ihnen die uneingeschränkte Nutzung der Vorzüge der neuen MS OS/2-Applikations-Software.«

Das Betriebssystem MS OS/2 für das MACH-20-System berücksichtigt 20 unterschiedliche Merkmale, die es zwischen dem PC-AT-Standard und dem PC-XT-Standard gibt. Einige dieser Unterschiede zwischen Systemen auf 80286-Basis und solchen auf 8088-Basis führen dazu, daß die Standardversion von MS OS/2 auf der früheren Hardware-Generation nicht läuft, auch wenn der 8088-Prozessor gegen einen 80286-Prozessor ausgetauscht wurde. Die Adaption des MS OS/2 für das MACH-20-System berücksichtigt alle diese Unterschiede. Die notwendigen Anpassungen wurden soweit wie möglich per Software vorgenommen, die zukünftige Aktualisierungen und Erweiterungen vereinfacht.

Mitteilungen .... Mitteilungen .... Mitteilungen

Neue Version des MS OS/2-Software-Development-Kit

Microsoft liefert jetzt die neueste Version des Software-Development-Kit für das Multitasking-Betriebssystem MS OS/2 aus. Das neue Release beinhaltet eine aktualisierte Version von MS OS/2, Release 1.1 - einschließlich »Presentation Manager« -, aktualisierte Entwicklungswerkzeuge wie den Font-Editor, Dialog-Editor und Icon-Editor sowie eine umfangreiche Dokumentation, zahlreiche Hilfen und Beispielprogramme, einen Online-Support über den Bildschirm und Trainings-Videocassetten.

»Die neue Version des Presentation Managers und die aktualisierten Entwicklungswerkzeuge bringen uns wieder einen Schritt weiter in Richtung auf das MS OS/2-Zeitalter«, so die Meinung von Christian Wedell, Geschäftsführer der deutschen Microsoft GmbH, zur Auslieferung der neuen Version des MS OS/2-Software-Development-Kits. »Mit Hilfe des neuen Releases können Entwickler von MS OS/2-Anwendungs-Software kontinuierlich an der neuen Generation von PC-Applikations-Software weiterarbeiten.«

Die neue Dokumentation wird durch ein elektronisches QuickHelp-Programm unterstützt, das als MS OS/2-Fenster läuft oder über die Befehlsebene aufgerufen werden kann. Die QuickHelp-Datenbank enthält das komplette Programmierer-Handbuch der MS OS/2-Version 1.1 sowie Beschreibungen der Entwicklungswerkzeuge des MS OS/2-Toolkits. Bis zum heutigen Tage sind bereits mehr als 4.500 MS OS/2-Software-Entwicklungs-Kits (SDK) weltweit durch Microsoft verkauft worden.

X/OPEN darf Netzwerk-Spezifikationen künftig veröffentlichen

Die X/OPEN-Gruppe und die Microsoft Corporation haben in diesen Tagen ein Abkommen unterzeichnet, das X/OPEN die Spezifikationen des Microsoft LAN-Managers für Unix bzw. den LAN-Manager/X zugänglich macht. Das Abkommen erlaubt es X/OPEN, diese Spezifikationen in zukünftigen Ausgaben des »X/OPEN Portability Guide« zu veröffentlichen. Zweck des »Portability Guide« ist es, sicherzustellen, daß Systeme auf der Basis der darin beschriebenen Spezifikationen ein konsistentes Interface haben, so daß die Portabilität und Ablauffähigkeit von Applikationssoftware auf unterschiedlichen Systemen gewährleistet ist.

Die Publizierung der LAN-Manager/X-Spezifikationen für die Protokolle und die Applikations-Programmierschnittstellen sollen sicherstellen, daß die Computer-Industrie zuverlässige Spezifikationen für Implementations- und Portierungszwecke erhält. Der LAN-Manager/X wird von Hewlett-Packard, einem führenden Mitglied der X/OPEN-Gruppe, und Microsoft gemeinsam entwickelt.

Geoff Morris, Präsident und CEO von X/OPEN meint zu dem mit Microsoft geschlossenen Abkommen, daß »der Markt sich an den Industrieführern orientiert, damit weniger Konfusion herrscht und eine Entwicklungsrichtung vorgegeben wird. In diesem Sinne stelle das Abkommen zwischen X/OPEN, der führenden Kraft in der Unix-bezogenen Standardisierung, und Microsoft, dem führenden Unternehmen im Bereich der PC-Software, ein konstruktives Element dar«. Christian Wedell, Geschäftsführer der deutschen Microsoft GmbH, ist der Meinung, daß »Microsoft mit der Lizenzierung der LAN-Manager-Spezifikationen an die X/OPEN-Gruppe seine positive Einstellung gegenüber offenen Systemen wieder einmal deutlich gemacht hat. Die Möglichkeit, PCs mit den beiden wichtigen Server-Plattformen MS-OS/2 und Unix zu verbinden, ist von großer Wichtigkeit für die Industrie.«

Der LAN-Manager für Unix ist eine auf Unix laufende Server-Software für lokale Netzwerke, die kompatibel mit dem MS-OS/2-Manager sein wird. Die Kombination von MS-OS/2-LAN-Manager und LAN-Manager/X schafft einen Standard und bietet einen zuverlässigen Weg, LAN-Serviceleistungen auf PC-Workstation verfügbar zu machen. Ein PC, der mit der LAN-Manager-Software arbeitet, wird in der Lage sein, einen transparenten Zugriff sowohl auf Unix als auch auf MS-OS/2 gestützte Server zu bieten. Der LAN-Manager kann darüber hinaus direkt Netzwerk-Software nutzbar machen, die bereits im Unix-Markt verbreitet ist, wie zum Beispiel die TCP/IP- und ISO-Protokolle.

Die X/OPEN-Gruppe ist ein Zusammenschluß der weltweit führenden Computer-Hersteller in den USA, Europa und Japan. Die Organisation dient der Schaffung und Veröffentlichung von Spezifikationen, die den Zweck haben, die Entwicklung offener Systeme zu ermöglichen und die Portabilität von Anwendungssoftware sicherzustellen. Mitglieder der X/OPEN-Gruppe sind die Unternehmen AT&T, Bull, Digital Equipment Corporation, Fujitsu, Hewlett-Packard, IBM, ICL, NCR, Nixdorf, Nokia, Olivetti, Siemens, Sun und Unisys.

3+Open-LAN-Manager, das erste auf dem MS-OS/2-LAN-Manager basierende OEM-Produkt

Das erste auf dem Microsoft-OS/2-LAN-Manager basierende OEM-Produkt ist auf den Markt: der 3+Open-LAN-Manager, entwickelt von der 3Com Corporation. Eric Benhamou, Vizepräzident und General Manager von 3Com, kommentierte die Auslieferung des neuen Produktes anläßlich des 3Com-Network-Systemforums mit den Worten: »Wir sind stolz, das erste Unternehmen zu sein, das eine Hochleistungs-Netzwerk-Software auf MS-OS/2-Basis auf den Markt bringt, die MS-DOS, MS-OS/2 und Macintosh-Workstations unterstützt«.

Mitteilungen

Messungen beim Einsatz der neuen Netzwerk-Software haben ergeben, daß der 3+Open-LAN-Manager erheblich schneller ist als die auf MS-DOS basierende 3+R-Netzwerk-Software von 3Com. Dieses Ergebnis wird durch Benchmarks der unabhängigen Firma Neal Nelson and Associates, Chicago, unterstrichen. Es sich zeigte dabei, daß der MS-OS/2-LAN-Manager eine bessere Leistung bei Standard-Applikationsprogrammen als vergleichbare Netzwerk-Software bringt und diese Leistung auch unter Last beibehält. Die Tests brachten außerdem zutage, daß der Microsoft LAN-Manager darüber hinaus ein exzellentes Leistungsverhalten in großen Netzwerken zeigt, in denen die einzelnen Systeme über Brücken verbunden sind.

Die Leistung eines Netzwerkes wird durch eine Vielzahl von Faktoren beeinflußt. Bei den bisher üblicherweise durchgeführten LAN-Benchmarks wurden in der Regel nur einzelne Komponenten des Systems gemessen, zum Beispiel sequentielle oder Random-Ein-/Ausgangs-Operationen. Ein wirklich aufschlußreicher Test einer Netzwerk-Betriebsumgebung sollte jedoch nicht auf die Geschwindigkeit einer einzelnen Operation beschränkt sein, sondern die Netzwerk-Leistung beim Einsatz einer echten kommerziellen Applikation aufzeigen. Der Neal-Nelson-LAN-Benchmark arbeitet deshalb mit Standard-Applikationen, wie »Lotus 1-2-3«, »Ashton-Tate dBase III Plus« und »Microsoft Word«, um die tatsächliche Leistungsfähigkeit einer LAN-Netzwerk-Systemsoftware zu ermitteln.

Auch die Firma DEC hat in diesen Tagen Pläne bekanntgegeben, die MS-OS/2-LAN-Manager-Technologie für den Einsatz auf ihren VAX-Minicomputern zu nutzen. DEC wird die LAN-Manager-Technologie einsetzen, um einen MS-OS/2- und DOS-Workstation-Support für ihre »VAX/VMS-Services for MS-DOS« zu bieten. VAX-Minis sollen zukünftig mit MS-OS/2-LAN-Manager-Servern zusammenarbeiten können.



Mitteilungen

Microsoft reorganisiert die Applications Division

Die Microsoft Corporation, Mutterfirma der deutschen Microsoft GmbH, gab in diesen Tagen eine weitreichende Neustrukturierung ihrer Applications Division in Geschäftsbereiche bekannt. Die Reorganisation dient dem Ziel, den Applikationsbereich von Microsoft auf das erwartete Wachstum in den nächsten fünf Jahren auszurichten. Jede der neuen »Business Units« hat einen zugeordneten Produktbereich und ist von der Konzeption über die Entwicklung bis hin zur Markteinführung und zur Produktpflege für die Produkte verantwortlich. Jeder Geschäftsbereich wird von einem General Manager geleitet, der an Mike Maples, Vice President of Applications, berichtet.

Wesentliche Aspekte der Umorganisation sind kürzere Kommunikationswege und eine stärkere Bündelung der jeweiligen Verantwortungsbereiche. Außerdem möchte man bei Microsoft die positiven Eigenschaften eines kleinen, flexiblen Unternehmens bewahren, obwohl die Firma sich zu einem Großunternehmen entwickelt.

Die neuen Geschäftsbereiche und ihre Geschäftsbereichsleiter sind:

Office Business Unit - mit den Produkten Microsoft Word und Microsoft Mail; Leitung: Jeff Raikes.

Graphics Business Unit - mit dem Produkt Microsoft PowerPoint; Leitung: Bob Gaskins.

Entry Business Unit - mit den Produkten Microsoft Works, Learning DOS und Flight Simulator; Leitung: Susan Boeschen.

Data Access Business Unit - mit den Produkten Microsoft File und Microsoft QuickBASIC; Leitung: Jeff Harbers.

Analysis Business Unit - mit den Produkten Microsoft Excel, Microsoft Project und Microsoft Multiplan; Leitung: Pete Higgins.

Entwicklungsunterstützung - Leitung: Peter Morse

Anwender-Interface-Architektur - Leitung: Tandy Trower.

Termine ... Termine ... Termine ... Termine

Mit Microsoft-Seminaren sicher in die Zukunft

Das Betriebssystem der Zukunft heißt Microsoft OS/2. Microsoft Windows und der Presentation Manager sind die Benutzeroberflächen der Zukunft sein. Für professionelle Entwickler bedeutet das, sich ab sofort mit dieser neuen Software auseinandersetzen zu müssen. Damit schaffen sie die Voraussetzung, schnellstmöglich Programme in der »neuen Welt« verfügbar zu haben.

Natürlich wird die Umstellung auf das neue Betriebssystem sowie die Programmerstellung nicht von heute auf morgen vollzogen sein. Um den Anfang jedoch so einfach wie möglich zu gestalten, bietet Microsoft eine Dienstleistung an: Das Microsoft Institut.

Die Spezialseminare des Microsoft Instituts vermitteln in kleinen Gruppen intensiv all das, was zum Einstieg in die Programmentwicklung nötig ist. Modernste Trainingsmethoden sowie PC-Demonstrationen und -Übungen sind selbstverständlich. Die Dozenten befassen sich auch im persönlichen Gespräch ausführlich mit den individuellen Forderungen und Problemen der Teilnehmer. So bekommen professionelle Entwickler durch professionelle Schulung die Möglichkeit, ihren hohen Wissenstand den neuen Gegebenheiten anzupassen.

Jeder Interessent in der Bundesrepublik Deutschland, der Schweiz und Österreich hat die Chance, sich mit der neuen Welt von Microsoft OS/2 und Microsoft Windows auseinanderzusetzen. Denn das Microsoft Institut arbeitet vor Ort mit kompetenten Schulungsunternehmen zusammen:

Digicomp AG, Zürich
Elektro-Calcul PI S.A., Lausanne
Integrata GmbH, Tübingen
INTEL Semiconductor GmbH, München
Olivetti Bildungs-Zentrum GmbH, Düsseldorf
Ueberreuter Media GmbH, Wien.

Die Dozenten werden speziell von Microsoft ausgebildet und stehen in ständigem Kontakt mit uns. So gibt es keine Informationsverluste: Das Wissen wird immer aktuell und aus erster Hand vermittelt. Microsoft erstellt die Seminare und die Seminarunterlagen und gewährleistet Qualität durch die Auswertung der Seminare.

Das Microsoft OS/2 Einführungs-Seminar

Das zweitägige Seminar wendet sich an PC-Software-Entwickler, die Programmierkenntnisse in einer höheren Programmiersprache wie C, Pascal, o.ä. besitzen.

Die Teilnehmer lernen im Vortrag und in Diskussionen das Konzept von Microsoft OS/2 kennen und erhalten einen Überblick über die Fähigkeiten und Programmierschnittstellen dieses Betriebssystems. Während des Seminars haben die Teilnehmer die Möglichkeit, das Gelernte anhand von Übungsaufgaben für sich selbst zu überprüfen.

Ort	Datum	Veranstalter
Düsseldorf	07./08.11.	OBZ
	05./06.12.	OBZ
Frankfurt	14./15.11.	Integrata
	05./06.12.	Integrata
Hamburg	12./13.12.	Integrata
München	24./25.10.	Integrata
	10./11.11.	Integrata
	19./20.12.	Integrata
Münster	28./29.11.	Integrata
Tübingen	07./08.11.	Integrata
	21./22.11.	Integrata
Linz	07./08.11.	Ueberreuter
Wien	29./30.11.	Ueberreuter
Lausanne	14./15.11.	Electro Calcul
Zürich	01./02.12.	Digicomp
	20./21.12.	Digicomp

Der Microsoft OS/2 Workshop

Das dreitägige Seminar wendet sich an PC-Software-Entwickler, die Programmiererfahrungen in einer höheren, strukturierten Programmiersprache unter MS-DOS und C-Kenntnisse besitzen sowie das MS-OS/2-Einführungsseminar besucht haben.

Die Teilnehmer lernen im Vortrag und praktischen Übungen am PC Family-API-Programme zu schreiben und Device-I/O-Routinen zu erstellen sowie Multitasking-Funktionen zu nutzen und eigene Dynamic-Link-Bibliotheken zu erstellen; außerdem können sie die erweiterten Speicherverwaltungsmöglichkeiten des Intel 80286 nutzen und mit Hilfe des MS-OS/2 Memory Managers programmieren. Dieses Seminar ist übrigens nicht im SDK-Preis enthalten.

Ort	Datum	Veranstalter
Düsseldorf	09./10./11.11.	OBZ
	07./08./09.12.	OBZ
Frankfurt	07./08./09.12.	Integrata
Hamburg	4./15./16.12.	Integrata
München	21./22./23.12.	Integrata
Münster	30./01./02.12.	Integrata
Tübingen	09./10./11.11.	Integrata
Lausanne	28./29./30.11.	Electro Calcul
Wien	14./15./16.12.	Ueberreuter
Zürich	12./13./14.12.	Digicomp

Termine ... Termine ... Termine ... Termine

Das Microsoft Windows Einführungs-Seminar

Das zweitägige Seminar wendet sich an PC-Software-Entwickler, die Programmierkenntnisse in einer höheren Programmiersprache wie C, Pascal, o.ä. besitzen.

Die Teilnehmer lernen im Vortrag und in Diskussionen das Konzept von Microsoft Windows kennen und erhalten einen Überblick über dessen Fähigkeiten und Programmierschnittstellen. Dieses Seminar ist nicht im SDK-Preis enthalten.

Ort	Datum	Veranstalter
Düsseldorf	21./22.11.	OBZ
	12./13.12.	OBZ
München	14./15.11.	Integrata
	05./06.12.	Integrata
	19./20.12.	Integrata
Münster	21./22.11.	Integrata
Tübingen	03./04.11.	Integrata
	28./29.11.	Integrata
	21./22.12.	Integrata
Salzburg	28./29.11.	Ueberreuter
Wien	14./15.11.	Ueberreuter
Zürich	10./11.11.	Digicomp
	05./06.12.	Digicomp

Microsoft Excel für Programmierer

Das dreitägige Seminar wendet sich an PC-Software-Entwickler, die Programmiererfahrung in einer höheren Programmiersprache haben.

Die Teilnehmer lernen im Vortrag und in praktischen Übungen am PC das Konzept und die Möglichkeiten, mit Microsoft Excel-Makros Applikationen zu erstellen.

Ort	Datum	Veranstalter
Düsseldorf	28./29./30.11.	OBZ
Münster	12./13./14.12.	Integrata
Genf	05./06./07.12.	Electro Calcul
Wien	07./08./09.11.	Ueberreuter
	05./06./07.12.	Ueberreuter

Microsoft Presentation Manager für Windows Programmierer

Ort	Datum	Veranstalter
Düsseldorf	28.11.88	OBZ
	19.12.88	OBZ
Wien	21.11.88	Ueberreuter

Der Microsoft Windows Workshop

Das dreitägige Seminar wendet sich an PC-Software-Entwickler, die Programmiererfahrungen in einer höheren, strukturierten Programmiersprache unter MS-DOS und C-Kenntnisse besitzen sowie das Microsoft Windows Einführungsseminar besucht haben.

Die Teilnehmer lernen im Vortrag und praktischen Übungen am PC Benutzerschnittstellen zu erstellen, die grafische Programmierschnittstelle zu nutzen, die Routinen zum Memory Management anzuwenden und dynamische Bibliotheken zu erstellen und zu benutzen. Dieses Seminar ist nicht im SDK-Preis enthalten.

Ort	Datum	Veranstalter
Düsseldorf	23./24./25.11.	OBZ
	14./15./16.12.	OBZ
Frankfurt	14./15./16.12.	Integrata
München	07./08./09.12.	Integrata
Münster	23./24./25.11.	Integrata
Tübingen	30./01./02.12.	Integrata
Zürich	28./29./30.11.	Digicomp

Multiplan 4.0:

Der Klassiker jetzt für MS-DOS und MS OS/2

Da ist sie also, die Version 4.0 des Tabellenkalkulationsprogramms Microsoft Multiplan, und läuft auch gleich unter OS/2. Somit bildet dieses Programm, das bei seiner Markteinführung 1982 die erste Standardanwendung des Sprachen- und Betriebssystemspezialisten Microsoft darstellte, nun ebenfalls die erste große Standardanwendung des Hauses Microsoft für das neue Betriebssystem MS OS/2.

Unabhängig davon bleibt Multiplan jedoch Bestandteil der Microsoft MS-DOS-Produktfamilie, die mit untereinander identischer Menüführung auch das Textverarbeitungsprogramm Microsoft Word, das Geschäftsgrafikprogramm Microsoft Chart und das Projektplanungsinstrument Microsoft Project umfaßt. Unverändert bleibt auch die Möglichkeit erhalten, Tabellen von Microsoft Multiplan komfortabel mit Microsoft Chart dreidimensional grafisch aufzubereiten. Den Programmpackungen von Multiplan liegen künftig je ein Diskettensatz für MS-DOS und für MS OS/2 bei. Der unveränderte Preis von 985 DM macht einmal mehr deutlich, daß der Um- oder Einstieg in den neuen MS OS/2-Standard nicht mit höheren Kosten verbunden ist.

Christian Wedell, Geschäftsführer der Microsoft GmbH, positioniert sein jüngstes Kind so: »Mit der neuen Version 4.0 haben wir Microsoft Multiplan vor allem zwei neue Einsatzfelder eröffnet: Tabellenkalkulation mit Datenbankfunktionen und Präsentation von Tabellen. Beide Aufgaben lassen sich sowohl im Multitasking unter dem Betriebssystem MS OS/2 als auch im Singletasking unter MS-DOS bewältigen. Unter den Tabellenkalkulationsprogrammen mit nichtgrafischer Oberfläche wird Microsoft Multiplan damit auch in Zukunft die technologische Spitzenstellung und mengenmäßige Marktführerschaft behaupten.«

Unter MS OS/2 lassen sich mit Microsoft Multiplan 4.0 mehrere unabhängige Tabellen simultan bearbeiten. Auch ein gleichzeitiger Zugriff auf MS OS/2-Datenbanken oder MS OS/2-Textverarbeitungsprogramme ist ohne Verlassen des Programms möglich.

Leistungsfähige Datenbankfunktionen

Während die Multitaskingfähigkeiten der neuen Version 4.0 nur unter MS OS/2 zum Tragen kommen, sind alle weiteren neuen Features auch unter MS-DOS verfügbar. Durch die Erweiterung um mehr als 30 neue Funktionen stehen nunmehr 95 verschiedene Spreadsheet- und Datenbankfunktionen zur Verfügung.

Gegenüber dem Vorgänger wurde Multiplan 4.0 vor allem um leistungsfähige Datenbankfunktionen ausgebaut. So lassen sich z.B. Datensätze wie eine Tabelle auf dem Arbeitsblatt der Tabellenkalkulation erfassen und auswerten. Dabei wird der Datenbankbereich wie bei Microsoft Excel mit einem spezifischen Namen definiert.

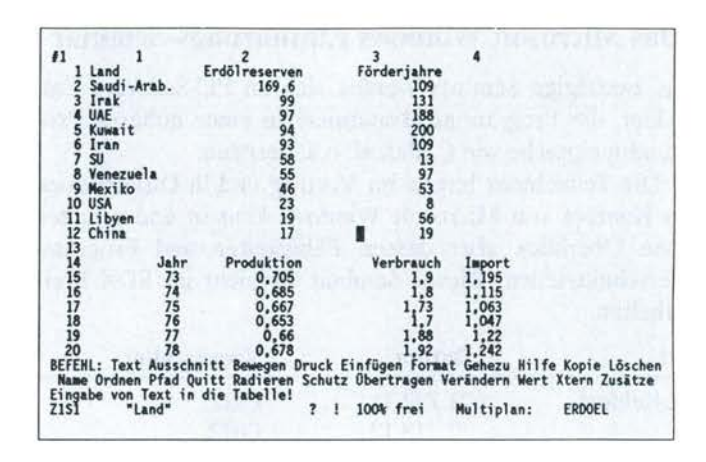


Bild 1: *Das gewohnte Erscheinungsbild von Microsoft Multiplan hat sich kaum verändert.*

Die Datenbank kann aus maximal 255 Feldern und 4095 Datensätzen bestehen, wobei die Daten direkt in die Tabelle eingegeben werden können. Eine vordefinierte Datenbankmaske ist im Unterschied etwa zu Microsoft Excel nicht vorhanden. Datenbanksätze können nach Kriterien selektiert werden. Dazu wird ein Suchkriterienbereich, der Feldbezeichnungen und Kriterien enthält, auf dem Arbeitsblatt definiert. Zur Selektion können einzelne Felder über »UND«- bzw. »ODER«-Verknüpfungen problemlos kombiniert werden. Innerhalb der Felder sind die gleichen Verknüpfungen erlaubt. Auch berechnete Suchkriterien sind zulässig. Mit der Funktion »FIND« wird der erste Datensatz markiert, der mit den definierten Suchkriterien übereinstimmt. Über die Cursortasten läßt sich die Markierung auf den vorhergehenden oder darauffolgenden Datensatz, der den Kriterien entspricht, setzen. Über den Befehl »EXTRACT« lassen sich alle Datensätze an eine beliebige Stelle im Arbeitsblatt kopieren. Dieser Befehl muß die Feldbezeichnungen enthalten, deren Inhalt ausgegeben werden soll. Der »EXTRACT«-Bereich wird dynamisch verwaltet, so daß eine zu kleine Auswahl des Bereichs ausgeschlossen wird. Bei der Befehlsausführung lassen sich Doppelnennungen unterdrücken.

Zur Auswertung des Datenbankbereichs wurden elf Datenbankfunktionen in Microsoft Multiplan 4.0 implementiert. Diese umfassen die Berechnung von Minimal-, Maximal- und Durchschnittswerten, Standardabweichungen, Populationsvarianten, Datenbanksummen und -produkten.

Das Maximum von 255 Spalten und 4095 Zeilen pro Tabelle verliert bei der Version 4.0 von Microsoft Multiplan dadurch an Bedeutung, daß Modelle auf mehrere Arbeitsblätter, die untereinander verknüpfbar sind, verteilt werden können. So lassen sich auch dreidimensionale Tabellen erstellen. Dabei sind bis zu acht verschiedene Tabellen gleichzeitig in unterschiedlichen Bildschirmfenstern darstellbar.

```
Database           Date            Financial       Logical
DBMITTELWERT()     DATUM()         GDA()           UND()
DBANZAHL()         DATWERT()       ZW()            FALSCH()
DBANZAHL2()        TAG()           ZINSZ()         WENN()
DBMAX()            STUNDE()        IKV()           ISTLEER()
DBMIN()            MINUTE()        QIKV()          ISTFEHL()
DBPRODUKT()        MONAT()         ZZR()           ISTFEHLER()
DBSTDABW()         JETZT()         BARWERT()       ISTLOG()
DBSTDABWN()        SEKUNDE()       RMZ()           ISTNV()
DBSUMME()          ZEIT()          KAPZ()          ISTZAHL()
DBVARIANZ()        ZEITWERT()      GW()            ISTPOS()
DBVARIANZEN()      WOCHENTAG()     ZINS()          ISTFOLGE()
                   JAHR()          LIA()           NICHTT()
                                   DIA()           ODER()
                                                   WAHR()

WERT: DBMITTELWERT()

DBMITTELWERT(D.bank;Feld;S.kriterien) M.wert der übereinstimmenden Felder!
Z31S2                    ? !  100% frei    Multiplan:    ERDOEL
```

Bild 2: *Auf Tastendruck (Shift F3) werden die Funktionen zur Auswahl angezeigt.*

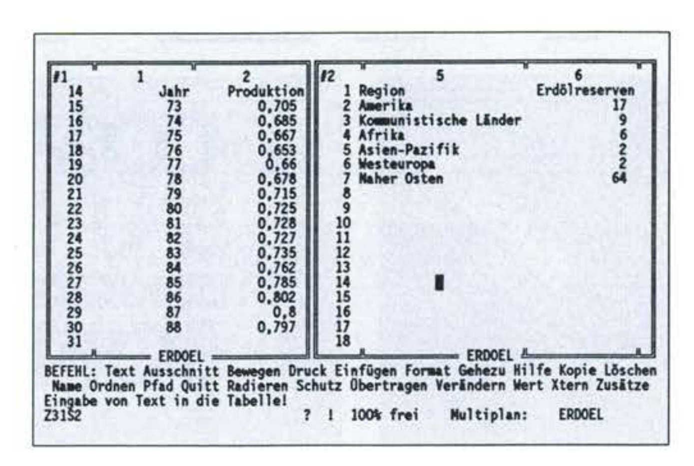


Bild 3: *Hier wurde die Darstellung mit Rahmen und zwei Fenstern gewählt.*

Zeilen oder Spalten können nach bis zu 20 verschiedenen Schlüsseln gleichzeitig in aufsteigender oder absteigender Folge sortiert werden.

Bedienungsfreundlicher und komfortabler

Für die neue Version 4.0 von Microsoft Multiplan wurden auch die Eingabemöglichkeiten für Formeln bedienungsfreundlicher gestaltet. So können jetzt die Funktionen mittels Tastendruck aus einer Liste verfügbarer Formeln eingefügt werden. In dieser Liste sind alle Funktionen nach Anwendungsgebieten geordnet aufgezählt. Wird die Markierung mit den Cursortasten oder der Maus auf eine der aufgelisteten Funktionen gesetzt, erfolgt in der Meldungszeile eine Kurzbeschreibung der entsprechenden Funktion.

Neu sind auch die verbesserten Gestaltungsmöglichkeiten von Tabellen. Einzelne Zeilen können mit der Version 4.0 von Multiplan nun in verschiedenen Farbdarstellungen am Bildschirm und am Drucker ausgegeben werden. So lassen sich z.B. - über Makros automatisiert - Negativwerte in Statistiken Rot, Positivwerte hingegen in Schwarz darstellen. Text kann fett, kursiv, unterstrichen oder durchgestrichen formatiert werden, wobei verschiedene Schriftarten zur Verfügung stehen. Damit diese Gestaltungsmöglichkeiten auch ausgegeben werden können, sind jetzt Druckertreiber vorhanden, die diese Merkmale unterstützen. Um Formatierungen bequem auszuwechseln, gibt es einen Befehl zum Suchen und Ersetzen von Formatangaben. Die maximale Spaltenweite beträgt 245 Zeichen. Die gewünschte Spaltenweite kann direkt im betreffenden Formatbefehl als Argument eingegeben oder mit den Cursortasten ausgedehnt werden. Durch die Eingabe des Buchstabens »d« läßt sich die Standardspaltenbreite wiederherstellen. Der Seitenrand kann nun wahlweise in Zeichen, Zoll oder Zentimetern bestimmt werden. Auch die Zahlenformate lassen sich frei definieren, wobei 24 unterschiedliche fest integriert sind - weitere lassen sich individuell hinzufügen.

Die vielen Neuerungen machten für Microsoft Multiplan 4.0 gegenüber der Vorgängerversion einige Änderungen im Befehlsmenü notwendig. Damit Makros aus älteren Anwendungen unverändert übernommen werden können, läßt sich per Befehl das Menü der Version 3.0 einstellen. Die Makrosprache selbst wurde um sechs Befehle erweitert. Es sind jetzt Befehle vorhanden, die die Anzeige von Aktionen in der Menüleiste und im Arbeitsblatt während des Makroablaufs unterdrücken oder aktivieren. Die damit verbundenen Vorteile: Durch Ausschalten der Bildschirmanzeige werden Makros wesentlich schneller und anwenderfreundlicher. Letzteres vor allem deshalb, weil nun selbst komplexeste Makroaktionen durchführbar sind, ohne daß der Anwender durch ständiges Bildschirmflackern irritiert wird. Darüber hinaus sind zwei Makrobefehle zum Editieren von Formeln hinzugekommen.

Problemloser Datenaustausch

Vor allem in großen Unternehmen, die Programme unterschiedlicher Hersteller einsetzen oder beim Umstieg von anderen Tabellenkalkulationsprogrammen auf Microsoft Multiplan, spielen Datentransfer-Möglichkeiten eine entscheidende Rolle. Dieser Funktionsbereich wurde in der neuen Version 4.0 von Microsoft Multiplan wesentlich erweitert. Es muß jetzt nicht mehr der Typ der zu übertragenden Daten manuell festgelegt werden - Microsoft Multiplan 4.0 erkennt automatisch, ob es sich um eine Lotus 1-2-3-, Lotus-Symphony-, ASCII- oder SYLK-Datei handelt. Auch der Datenaustausch mit Programmen, die unter Microsoft Windows ablaufen, wurde erheblich erleichtert. Microsoft Multiplan 4.0 ist in der Lage, Dateien im ASCII- oder SYLK-Format aus der Zwischenablage - dem Clipboard - von Windows zu laden oder sie direkt in die Zwischenablage zu speichern.

Daten und Fakten im Überblick

Microsoft Multiplan 4.0 ist ablauffähig auf Rechnern mit MS-DOS 2.0 oder höher bzw. auf MS OS/2 Version 1.0, die über zwei Disketten- oder ein Disketten- und Festplattenlaufwerk verfügen. Unter MS OS/2 wird der Protected Mode unterstützt. Nötig sind mindestens 384 Kbyte Hauptspeicher. Es werden Netzwerke auf der Basis von MS-Net oder des Microsoft OS/2 LAN Managers unterstützt. Für registrierte Besitzer der Vorgängerversion 3.0 bietet Microsoft ein vollständiges Update zum Preis von 296 DM an.

Datenbank

Um Listen besser auswerten zu können und nach bestimmten Informationen zu suchen, bietet Microsoft Multiplan 4.0 die Möglichkeit, eine Tabelle oder Teile dieser als Datenbank zu organisieren. Mit Befehlen können Daten nach bestimmten Kriterien gesucht und extrahiert werden. Außerdem weist Microsoft Multiplan 4.0 statistische Funktionen für die Arbeit mit der Datenbank auf.

OS/2-Unterstützung

Multiplan 4.0 ist jetzt auch unter dem Betriebssystem MS OS/2 im OS/2-Modus ablauffähig. Das bedeutet, daß Multiplan gleichzeitig mit anderen Anwendungsprogrammen ausgeführt oder mehrmals gestartet werden kann, um die verschiedensten Aufgaben zu bearbeiten.

Verbesserte Feldformatierung

Microsoft Multiplan 4.0 ermöglicht eine Hervorhebung in Feldern. Die Zeichenformate Fett, Unterstrichen und Kursiv stehen zur Verfügung.

Funktionsauswahl

Multiplan 4.0 weist mehr als 30 neue Funktionen auf, darunter Datenbankfunktionen, finanzmathematische, mathematische und statistische Funktionen, sowie Sonder- und Textfunktionen, die die Formelerstellung erleichtern. Am Bildschirm kann man in einer Liste alle Funktionen nachsehen und diese in Formen einfügen, ohne im Handbuch nachzuschlagen.

Änderungen am Bildschirm

Der Bildschirm von Multiplan 4.0 weist mehrere Verbesserungen auf:

- Eine neue Statuszeilenanzeige erscheint: EW für den Modus »Erweiterung« (wenn vor der Wahl eines Befehles ein Bereich markiert wurde).
- Bei einer Verbindung mit einem externen Programm zeigt Multiplan die Gesamtzahl der zu kopierenden Felder an und zählt dann bis Null zurück, damit der Ablauf nachverfolgt werden kann.
- Für Mausbenutzer steht ein »!« neben dem »?« Symbol. Wie das Drücken von [F4] bewirkt Anklicken des »!«, daß die Werte in der Tabelle neu berechnet werden.
- Horizontale und vertikale Seitenumbrüche können als Teil der Ausschnittumrahmung angezeigt werden.

Vorauswahl eines Tabellenbereiches

Für welchen Bereich ein Befehl ausgeführt werden soll, kann man sich einfacher vorstellen, wenn der entsprechende Bereich vor der Befehlsauswahl markiert wird. Markiert man also den gewünschten Teil einer Tabelle und wählt danach einen Befehl aus, so erscheint der zuvor hervorgehobene Bereich als Vorgabe in den entsprechenden Befehlsfeldern. Für Befehle, die keine ausdrücklichen Feldpositionsangaben erfordern, zum Beispiel Einfügen und Löschen, gibt Multiplan eine Vorgabe ein, die für den zuvor markierten Bereich geeignet ist. Die Feldauswahl kann auch durch Ziehen der Maus erfolgen.

Einstellen der Druckoptionen

Die Ränder auf ausgedruckten Tabellen lassen sich in Anzahl der Zeichen (`Ze`), Zoll (`In`) oder Zentimeter (`Cm`) angeben.

Ordnen

Beim Ordnen von Ziffern oder Spalten wird jede angegebene Zeile bzw. Spalte als ein separater Sortierschlüssel behandelt, wobei die erste Zeile bzw. Spalte in der Liste die wichtigste ist. Microsoft Multiplan bietet in diesem Befehl die Möglichkeit, nach bis zu 20 Schlüsseln gleichzeitig zu sortieren.

Änderungen zum Befehl Übertragen Laden

Der Befehl `Übertragen Laden` ist jetzt kontextabhängig: d.h. Multiplan 4.0 lädt Tabellen und Dateien, die im Multiplan-, SYLK- oder ASCII-Format gespeichert sind, automatisch in dem Format, das im Befehl `Übertragen Optionen` zuvor festgelegt wurde.

Speichern eines Teiles einer Tabelle

Teile einer Tabelle lassen sich durch Angabe des Speicherbereiches im Befehl Übertragen Optionen speichern, um z.B. eine sehr große Datei in übersichtlichere Tabellen zu splitten.

Microsoft Windows-Zwischenablage

Tabellen oder Teile von Tabellen, die im ASCII- oder SYLK-Format gespeichert wurden, lassen sich in der Windows-Zwischenablage speichern und von dort laden.

Darstellung des Befehlsbereiches

Die Farbe von Befehlsnamen kann zur Unterscheidung von Befehlsnamen und Optionen geändert werden.

Makros

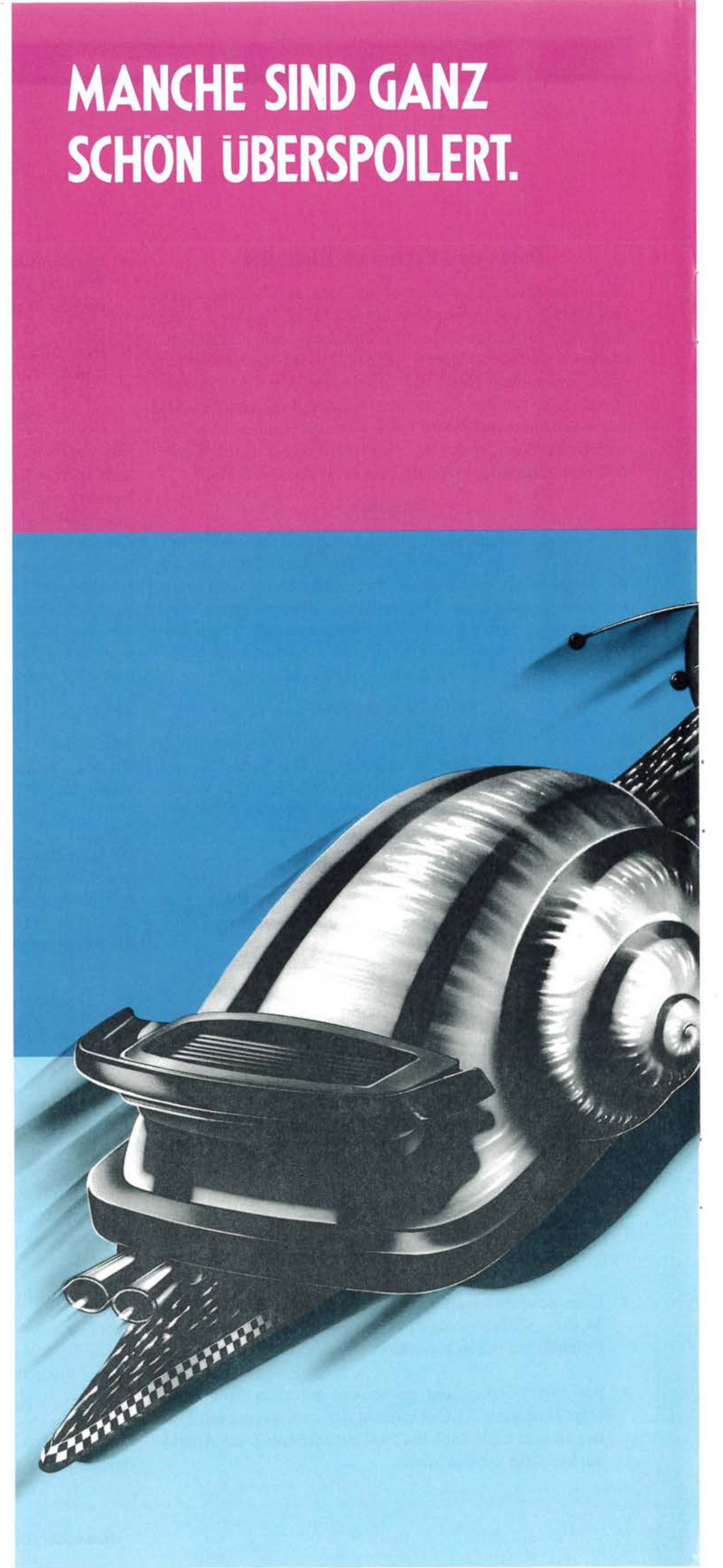
Multiplan 4.0 weist mehrere neue Makrobefehle auf, die das Aus- und Einschalten der Menü- und Bildschirmaktualisierung ermöglichen und das Bearbeiten von Eingaben in Befehlsfeldern erleichtern. Die neuen Makrobefehle lauten:

`'AM`
: schaltet das Menü aus. Bei Verwendung dieses Makrobefehls aktualisiert Multiplan die Menüs während des restlichen Makros nicht, bzw. so lange nicht, bis das Programm auf das Makro `'EM` oder ein anderes Makro stößt, das eine Eingabe wie zum Beispiel `'MF` oder `'IS` erfordert. Dadurch kann Multiplan komplexe und visuell beeindruckende Makros durchführen, ohne daß der Anwender durch das Aktualisieren des Menüs abgelenkt wird. Im Einzelschrittmodus ignoriert Multiplan diesen Makrobefehl.

`'EM`
: schaltet die Menüs ein. Dieser Befehl wird während eines Makros nach Verwendung des Makrobefehls `'AM` zum erneuten Einschalten der Menüaktualisierung verwendet.

`'AB`
: schaltet den Bildschirm aus. Bei Verwendung dieses Makrobefehls aktualisiert Multiplan den Bildschirm während des restlichen Makros nicht, bzw. so lange nicht, bis das Programm auf das Makro `'EB` oder ein anderes Makro stößt, das eine Eingabe wie zum Beispiel `'MF` oder `'IS` erfordert. Dieser Makrobefehl ist dem `'AM`-Makro ähnlich. Im Einzelschrittmodus ignoriert Multiplan diesen Makrobefehl.

`'EB`
: schaltet den Bildschirm ein. Dieser Befehl wird während eines Makros



nach Verwendung des Makrobefehls `'AB` zum erneuten Einschalten der Bildschirmaktualisierung verwendet. Multiplan aktualisiert den Bildschirm immer dann, wenn das Programm auf diesen Makrobefehl stößt.

`'HE`

geht zum Anfang der zu bearbeitenden Eingabe in einem Befehlsfeld. Dieses Makro entspricht dem Drücken vom [Shift][F7].

`'EN`

geht zum Ende der zu bearbeitenden Eingabe in einem Befehlsfeld. Dieser Makro entspricht dem Drücken von [Shift][F8].

Ändern der Feldbreite

Jetzt lassen sich die Richtungstasten zum Ändern der Feldbreiten in den Menüpunkten `Format Breite der Spalten` und `Format Standard Breite der Spalten` verwenden. Mit jedem Tastendruck ändert sich die Spaltenbreite direkt am Bildschirm, so daß die Änderungen direkt überprüfbar sind.

Kompatibilität mit alten Menüs

Die Befehlsmenüs wurden in Multiplan 4.0 zum Teil geändert. Trotzdem lassen sich unter Microsoft Multiplan 3.0 erstellte Makros problemlos einsetzen. Denn Microsoft Multiplan 4.0 erlaubt es, das Menü der Version 3.0 anzeigen zu lassen.

Druckertreiber

Multiplan 4.0 kann Tabellen jetzt einfach gestalten und formatieren, ohne daß Druckersteuerzeichen eingegeben werden müssen. Man wählt einfach, wie von Microsoft Word gewohnt, den gewünschten Druckertreiber aus. Und schon lassen sich Tabellen mit den Schriftarten und -größen versehen, die in einem Menü am Bildschirm angeboten werden.

pi

Das System Journal versucht seine Bücherflut zu bewältigen:

Bücher über Bücher

In den letzen Monaten hat sich eine solche Fülle interessanter Bücher zu Programmierthemen in der Redaktion angesammelt, daß wir sie diesmal nicht in der gewohnten Ausführlichkeit besprechen können. Deshalb bringen wir hier zum großen Teil nur Kurzbesprechungen. Wir hoffen, beim nächsten Mal wieder ausführlicher sein zu können.

Windows

Es ist soweit: Endlich gibt es auch ein deutsches Buch über das *»Programmieren unter MS-Windows«*. Es ist im Hanser-Verlag erschienen und stammt von Jacek Skarbek und Herbert Thiele. Es handelt sich dabei um ein Lehr- und Arbeitsbuch über die Benutzung der Funktions-Bibliotheken und Begriffe des Windows Software Development Toolkits, das vor allem die ersten Schritte in die Windows-Umgebung erleichtern soll. Anhand zahlreicher Anwendungsbeispiele wird besonders auf die Selbsterstellung von Anwendungsprogrammen eingegangen, die die Besonderheiten von Windows nutzen, die grafische Benutzeroberfläche und die Multitasking-Fähigkeiten, und Text- und Grafikinformationen austauschen. Es ist zu hoffen, daß das Buch der Anfang einer Reihe von hoffentlich ebenso soliden deutschen Windows-Büchern sein wird.

Jacek Skarbek, Herbert Thiele: »Programmieren unter MS-Windows«, München: Hanser, 1988; 375 Seiten; ISBN 3-446-15173-7; DM 78,-.

QuickBasic

Die interessanteste Neuerscheinung über QuickBasic ist die *»Microsoft QuickBasic Programmer's Toolbox«* von John Clark Craig. Dieses Buch besteht fast nur aus Listings mit kurzen Beschreibungen. Insgesamt werden über 250 Unterprogramme und Funktionen präsentiert, die sich über alle Bereiche der Programmierung erstrecken, von der Bildschirm- und Maussteuerung über String- und Bitmanipulationen bis zu Statistik- und Kalenderfunktionen. Neben der Verwertbarkeit ist von besonderem Nutzen, daß man sieht, wie schön man mit QuickBasic modular und strukturiert programmieren kann. Gesamturteil: Sehr empfehlenswert.

Doch auch die deutsche QuickBasic-Bücher haben etwas zu bieten. Gerd Kebschull beschreibt in seinem QuickBasic-Handbuch die Versionen 2.01 bis 4.0 von QuickBasic, wobei auch auf die Unterschiede eingegangen wird. Einen großen Teil des Buches macht eine Beschreibung der Befehle und Funktionen von QuickBasic aus, wobei in der Regel mehrere Beispiele geboten werden. In einer 40seitigen Programmsammlung werden einige komplette Programme vorgestellt, von der Musikerzeugung bis zum Spiel.

John Clark Craig: »Microsoft QuickBasic Programmer's Toolbox«, Redmond: Microsoft Press, 1988; 500 Seiten; ISBN 1-55615-127-6; $22.95. Diskette $23.95.

Gerd Kebschull: »Das QuickBASIC Handbuch«, Düsseldorf: Sybex, 1988; 652 Seiten; ISBN 3-88745-523-1; DM 49,-.

QuickC und Microsoft C

Daß sich Microsoft Press mit seinen Bücher außergewöhnlich viel Mühe gibt, zeigt auch wieder das Buch *»Microsoft QuickC Programming«* von der Waite Group. Man hat sich mit Mitchel Waite und Stephen Prata zwei Autoren geholt, die sich mit qualitativ hochwertigen Büchern über die Programmiersprache C in anderen Verlagen einen Namen gemacht haben, und damit ein Buch bekommen, daß derzeit das wohl beste in dieser Kategorie ist. Wer englischsprachige Fachbücher lesen kann und sich für QuickC interessiert, sollte dieses Buch ganz oben auf seine Einkaufsliste setzen.

Wer Deutsch vorzieht, erhält mit *»Programmieren mit Quick C«* von Rainer Haselier und Klaus Fahnenstich eine gute Einführung in das Thema, die vor allem davon profitiert, daß die Autoren zahlreiche Schulungen durchgeführt haben. Die Beispielprogramme werden, wie das beim Verlag Markt & Technik erfreulicherweise inzwischen schon Standard ist, auf zwei Disketten mitgeliefert.

Said Baloui, der sich mit seinen Tool-Disketten zu QuickBasic einen Namen gemacht hat, hat nun unter dem Titel *»Profi-Tools QuickC«* auch eine Tool-Sammlung für QuickC herausgebracht. Auch hier gehört die Diskette zum Lieferumfang. Beschrieben werden in gewohnter Qualität zahlreiche Unterroutinen, die als Grundlage für jedes Programm benötigt werden, von der Stringbehandlung bis zur Fenster- und Menüverwaltung.

Um eine Loseblattsammlung im DIN-A4-Format handelt es sich bei *»Erfolgreiches Programmieren mit komfortablen Musterlösungen in C«*, das von Jürgen Riederer herausgegeben wurde. Der Vorteil einer Loseblattsammlung liegt auf der Hand, sie ist preiswert erweiterbar und leicht auf dem neuesten Stand zu halten. So erscheinen hier alle 2 bis 3 Monate Ergänzungsausgaben mit einem Umfang von ca. 120 Seiten und einer Diskette zum Preis von DM 59,-. Neben der Einführung in die Sprache C bietet diese Sammlung auch viele Programmbeispiele, deren Glanzstück ein 8086-C-Compiler ist.

The Waite Group: »Microsoft QuickC Programming; The Microsoft Guide to Using the QuickC Compiler«, Redmond: Microsoft Press, 1988; 607 Seiten; ISBN 1-55615-048-2; $19.95. Diskette $22.95.

Rainer Haselier, Klaus Fahnenstich: »Programmieren mit Quick C«, Haar: Edition Microsoft im Markt & Technik Verlag, 1988; 412 Seiten; inkl. 2 Disketten; ISBN 3-89090-609-5; DM 69,-.

Said Baloui: »Profi-Tools Quick C«, Haar: Markt & Technik, 1988; 209 Seiten; 1 Diskette; ISBN 3-89090-692-3; DM 98,-.

Jürgen Riederer: »Erfolgreiches Programmieren mit komfortablen Musterlösungen in C«, Kissing: Interest-Verlag, 1988; Bestell-Nr. 3600; DM 98,-. Diskette (Grundwerk-Progr. und 8086-C-Compiler) DM 39,90. Ergänzungslieferungen DM 59,-.

Assembler

Auch für die Assemblerprogrammierung bietet der Interest-Verlag mit dem *»Intel 16 Bit Assemblerhandbuch«* von Jobst, Lutz und Selder eine Loseblattsammlung an. Sie vermittelt nicht nur Assembler- und Hardwarekenntnisse, sondern enthält auch Module eines 8086-Makroassemblers. Schritt für Schritt kann man hier Entwicklung und Aufbau des am MASM-Standard ausgerichteten Programmierwerkzeugs mitverfolgen, das in den Ergänzungsausgaben laufend erweitert wird.

Ein Standard für den Einstieg in die Assemblerprogrammierung auf dem amerikanischen Markt ist jetzt auch auf deutsch erhältlich: *»Peter Norton's Assemblerbuch«*. Es ist zwar in den USA allgemein bekannt, daß Peter Norton zu diesem Buch wenig mehr als seinen Namen beigetragen hat, doch das hat dem Erfolg des Buches keinen Abbruch getan, denn auch John Socha ist als Entwickler solcher Programme wie dem Norton Editor oder dem Norton Commander kein unbeschriebenes Blatt. Zu empfehlen ist das Buch wegen seines hervorragenden didaktischen Aufbaus vor allem dem Assembleranfänger. Sehr schön ist auch, daß das Ziel der Entwicklung ein umfangreiches und allgemein nützliches Disk-Patch-Programm ist, dessen Programm-Module sich auch als Basis für andere Programme eignen.

Aber auch weniger bekannte Autoren können gute Bücher über die Assemblerprogrammierung schreiben, wie Peter Monadjemi mit seinem Buch *»PC-Programmierung in Maschinensprache«* zeigt. Hier wird jedoch stärker auf Besonderheiten wie die Programmierung von Grafikausgaben auf der Herkuleskarte eingegangen und auch umfangreichere Informationen zum Nachschlagen geboten, z.B. ausführliche Beschreibungen aller Assemblerbefehle oder Interruptübersichten. Für Anfänger vorteilhaft sind die Übungen am Ende jedes Kapitels, die zusammen mit den Lösungen im Anhang eine effektive Kontrolle des Lernvorgangs erlauben.

Jobst/Lutz/Selder: »Intel 16 Bit Assemblerhandbuch; Programmiertechnik und Programmsammlung für IBM PC's und Kompatible«, Kissing: Interest-Verlag, 1987; Bestell-Nr. 2200; DM 98,-. Diskette 1 (Grundwerks-Programme und ASM) DM 39,90; Diskette 2 (Programme der 1. und 2. Ergänzung und SIG.ASM) DM 39,90. Ergänzungslieferungen DM 59,-.

Peter Norton, John Socha: »Peter Norton's Assemblerbuch«, Haar: Markt & Technik, 1988; 418 Seiten; inkl. 2 Disketten; ISBN 3-89090-624-9; DM 79,-.

Peter Monadjemi: »PC-Programmierung in Maschinensprache«, Haar: Markt & Technik, 1988; 623 Seiten; inkl. 1 Diskette; ISBN 3-89090-503-X; DM 69,-.

Problemlösungen

Robert Jourdain liefert mit seinem Buch *»Der PC-Problemlöser«* eine Nachschlagewerk, das gleichzeitig als Anregung für die Lösung von typischen Programmierproblemen auf PCs gedacht ist. Zu jedem Problem wird ein Beispielprogramm geliefert, wodurch dem Leser eben nicht nur die nötigen Informationen geboten werden, wie das von einem Nachschlagewerk erwartet wird, sondern auch gleich eine lauffähige Lösung. Man spart dadurch sehr viel Programmier- und Testzeit. Jedes Problem wird auf verschiedenen Ebenen gelöst (obere, mittlere, untere). Auf der oberen Ebene wird erläutert, wie das Problem in einer höheren Programmiersprache zu lösen ist (meist Basic), bzw. welcher Befehl dafür verfügbar ist. Für die beiden anderen Ebenen werden überwiegend Beispiele in Assembler beschrieben.

Eine stichwortartige Übersicht der behandelten Themen zeigt die Vielfalt der Problemlösungen: Systemausstattung feststellen, Interruptbehandlung, Speicherverwaltung, Zeitgeber, Tonerzeugung, Abfrage und Programmierung der Tastatur, Bildschirmausgaben, Pixelgrafik, Rollen und Blättern, Datei- und Verzeichnisoperationen, Druckvorgänge, Sonderzeichen, Hardcopies, Programmierung der Schnittstellen, Einheitentreiber u.v.m.

Das Buch wird damit seinem gesetzten Ziel gerecht: Es dürfte darin jeder Programmierer zahlreiche Hinweise finden, die ihm die Lösung eigener Programmierprobleme erleichtern.

Robert Jourdain: »Der PC-Problemlöser / Die 144 häufigsten Programmierprobleme und ihre Lösungen«, München: Hanser, 1988; 510 Seiten; ISBN 3-446-15348-9; DM 78,-.

Robert Jourdain ist auch der Ko-Autor von *»Peter Norton's Hard Disk Kompendium«*. Hier geht es jedoch nicht um Programmierprobleme, sondern um die Beschreibung von Festplatten und ihren besonderen organisatorischen Erfordernissen. Viele Anwender halten Festplatten immer noch für gigantische Disketten, was dann in der Praxis zu vielen Problemen führen kann, von der eingeschränkten Dateianzahl im Hauptverzeichnis bis zum Vergessen der notwendigen Datensicherung. Das Hard-Disk-Kompendium zeigt, wie Festplatten aufgebaut sind, wie Daten darauf strukturiert werden können, wie man geeignete Platten auswählt, welche Software ratsam ist, wie sie installiert werden, wie sich die Performance verbessern läßt, welche Sicherungsarten möglich und zu empfehlen sind und wie man Festplatten-Katastrophen mit möglichst geringem Datenverlust übersteht. Alles Themen, die nicht nur für »normale Sterbliche«, sondern auch für Programmierer interessant sind. Besonders die Kapitel über die Datensicherung seien jedem ans Herz gelegt. Es ist erstaunlich, wie oft man immer noch von totalem Datenverlust gerade auch bei Programmierern hört, weil die Datensicherung viel zu oft noch als lästiges und überflüssiges Übel angesehen wird.

Peter Norton, Robert Jourdain: »Peter Norton's Hard Disk Kompendium«, Haar: Markt & Technik, 1988; 459 Seiten; ISBN 3-89090-645-1; DM 79,-.

Ein Kapitel aus »Video Systems« von Richard Wilton:

Bildschirmausdruck in Assembler

Obwohl alle Mitglieder der IBM PC- und PC/2-Familie mit einer ROM BIOS-Routine ausgestattet sind, die die Ausgabe des aktuellen Bildschirmpufferinhalts auf einem Drucker ermöglicht, kann es doch notwendig werden, ein eigenes Programm zur Ergänzung oder Ersetzung der Bildschirmroutine zu schreiben. Dieser Buchauszug beschäftigt sich mit solchen Bildschirmdruckroutinen und mit der Frage, wie und warum ein eigenes Programm geschrieben werden kann.

Alphanumerische Modi

Die für den Bildschirmausddruck zuständige Routine befindet sich auf der Hauptplatine im ROM und wird über den Softwareinterrupt 5 aufgerufen (der Tastatur-Handler führt diesen Interrupt aus, wenn die Tastenkombination [Shift] [PrtSc] betätigt wird). Auch innerhalb eines Programm kann der INT 5 ausgeführt werden. Die Routine kopiert den Inhalt der aktuellen Bildschirmseite zum Drucker und verwendet dazu die alphanumerischen Modi mit einer Auflösung von 80x25 oder 40x25 Zeichen. Die Routine druckt lediglich die ASCII-Zeichencodes, die Attributbytes im Bildschirmpuffer werden ignoriert.

EGA, MCGA, VGA

Das ROM BIOS der EGA, MCGA und VGA verfügt über eine flexiblere Version der Routine INT 5. Diese verwendet den Wert ROWS (0040:0084) im Bildschirmdatenanzeigebereich, um die Anzahl der zu druckenden Zeichenzeilen zu bestimmen (die ROM-Version auf der Hauptplatine dagegen druckt immer 25 Zeilen). Die Hauptplatinenversion wird beim IBM PC/XT oder IBM PC/AT standardmäßig verwendet. Um die ROM BIOS-Routine von EGA oder VGA über Interrupt 5 zugänglich zu machen, muß der Interrupt INT 10H mit BL = 20H aufgerufen werden. Dadurch wird der Vektor auf die flexiblere Routine gesetzt.

Blockgrafikzeichen

Da die meisten Drucker für das Arbeiten mit verschiedenen Computern ausgelegt sind, ist nicht immer der gesamte ASCII-Zeichensatz verfügbar. Besonders die Zeichen, die für den Einsatz in Blockgrafiken vorgesehen sind, sind (leider auch) auf IBM-kompatiblen Druckern nicht immer vorhanden. Die Zeichen oberhalb des ASCII-Codes 128 werden mitunter mit einem anderen Erscheinungsbild oder gar nicht ausgedruckt.

Grafikmodi

Das ROM BIOS unterstützt keinen Bildschirmausdruck im Grafikmodus, so daß in diesen Modi eigene Programme für den Ausdruck des Bildschirmpufferinhalts herangezogen werden müssen.

GRAPHICS

Bei GRAPHICS handelt es sich um ein RAM-residentes Programm für den Bildschirmausdruck. Dieses Programm wird von Microsoft als Teil des Betriebssystems MS-DOS geliefert und trägt die Bezeichnung GRAPHICS.COM oder GRAPHICS.EXE. Bei der Ausführung von GRAPHICS.EXE wird ein speicherresidentes Programm für die CGA-Grafikmodi (320x200 mit vier Farben und 640x200 mit zwei Farben) in den Hauptspeicher geladen. Das Programm wurde für die Ausgabe auf einem IBM- oder Epson-kompatiblen Drucker entwickelt.

Der RAM-residente Teil von GRAPHICS fängt Interrupt 5 ab und überprüft den aktuellen Bildschirmmodus. Falls sich der Adapter in einem Grafikmodus befindet, wird ein Bildschirmausdruck erstellt, andernfalls übernimmt die BIOS-Routine des Interupt 5 den Bildschirmausdruck im alphanumerischen Modus. Ein Bildschirmausdruck im Grafikmodus kann durch das Betätigen der Tastenkombination [Shift][PrtSc] erhalten werden, so wie das auch in den alphanumerischen Bildschirmmodi der Fall ist.

Entwicklung einer Routine für den Bildschirmausdruck

Falls auf einer EGA-, VGA-, MCGA- oder Herkules-Grafikkarte ein Ausdruck gewünscht wird oder GRAPHICS auf dem Drucker nur unzulängliche Ausdrucke produziert, kann eine eigene Routine für den Bildschirmausdruck entwickelt werden. *Listing 1* gibt für die CGA-Grafikroutinen ein einfaches Beispiel. Die Routine `ScreenDumpCGA` läßt sich in ein Assemblerprogramm oder in ein Programm einer anderen höheren Programmiersrache integrieren, indem es mit den entsprechenden Registerwerten im richtigen Speichermodell aufgerufen wird. `ScreenDumpCGA` kann auch in ein speicherresidentes Programm eingebaut werden, das wie GRAPHICS den Interruptvektor 5 umleitet und immer dann ausgeführt wird, wenn die Tastenkombination [Shift][PrtSc] im Grafikmodus gedrückt wird.

`ScreenDumpCGA` kopiert Pixel aus dem Bildschirmpuffer in einen Zwischenpuffer und formatiert diesen so, daß der Inhalt direkt an den Drucker übergeben werden kann (in diesem Beispiel wird mit einem Epson MX-80 gearbeitet). Da auf den Bildschirmpuffer wahlfrei zugegriffen werden kann, liest `ScreenDumpCGA` die Pixel in der Reihenfolge, in der sie an den Drucker gesendet werden.

Das Herzstück von `ScreenDumpCGA` ist die Unterroutine `TranslatePixels`. Diese Routine bildet Pixel aus dem Bildschirmpuffer in den Druckerpuffer ab. In diesem Fall handelt es sich um eine kurze und schnelle Routine, da die zur Umwandlung von Bildschirm- in Druckerpixel eingesetzte Transformationsvorgabe recht einfach ist, da der MX-80 vertikal orientierte Pixel ausdruckt (siehe *Bild 1*). Die einfachste Methode, um Pixel aus einem horizontal ausgerichteten Bildschirmpuffer auszudrucken, ist die Drehung um 90 Grad.

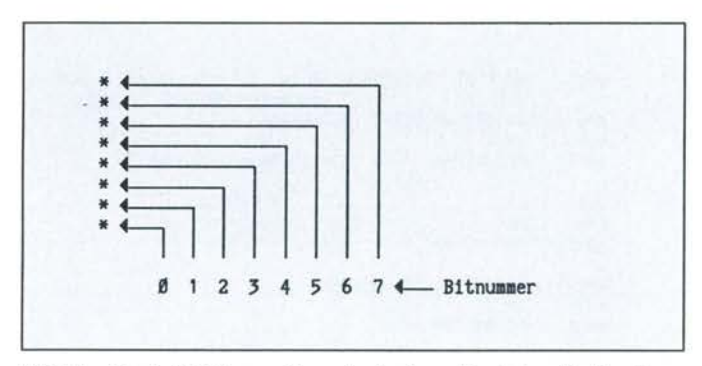


Bild 1: *Pixelabbildung eines typischen Punktmatrixdruckers. Bei der Verschiebung des Druckkopfes über die Seite werden gleichzeitig acht Pixelzeilen gedruckt. Jedes an den Drucker übergebene Byte beschreibt acht vertikal angeordnete Pixel, wie das Bild zeigt.*

Die Anpassung der Routine `ScreenDumpCGA` konzentriert sich darauf, wie Pixel aus dem Bildschirmpuffer am besten in den Druckerpuffer abgebildet werden. Dazu kann in `TranslatePixels` die Skalierung oder Rotation der Pixel geändert werden, oder es wird die Routine `ScreenDumpCGA` modifiziert, und so die Reihenfolge geändert, in der der Inhalt vom Bildschirm- in den Druckerpuffer kopiert wird.

Beispiel: `ScreenDumpCGA` und `TranslatePixels` können so modifiziert werden, daß der Inhalt des EGA-Bildschirmpuffers (oder des VGA) im 640x350-Farbmodus (16-farbig) ausgedruckt werden kann. Die Routine in *Listing 2* druckt alle Pixel mit dem Wert 0 als schwarze Punkte. Beachten Sie, wie der Lesemodus 1 des Controllers diese Aufgabe in `TranslatePixels` vereinfacht.

RAM-gestützte alphanumerische Zeichendefinitionen

Auch die Routine für den Bildschirmausdruck im Grafikmodus kann so modifiziert werden, daß RAM-gestützte Zeichen ausgedruckt werden, die in den alphanumerischen Modi des EGA, MCGA, VGA, HGC+ und der InColor Karte verwendet werden. Dies ist möglich, wenn die im Bildschirmpuffer gespeicherten Zeichencodes zur Indizierung des Zeichendefinitions-RAM eingesetzt werden. Das Bitmuster, durch das jedes Zeichen definiert ist, kann als Punktmuster für den Drucker eingesetzt werden.

In *Listing 3* wird dies an den Zeichen, die in den Zeichendefinitionstabellen des EGA bzw. VGA definiert sind, gezeigt. Die Routine druckt jede Zeichenspalte im Bildschirmpuffer, indem der Puffer (`PrintBuf`) mit den Bitmustern, die die Zeichen definieren, gefüllt wird. Die Memory-Maps 0 (enthält Zeichencode) und 1 (enthält Zeichendefinition) werden in der Unterroutine `TranslatePixels` getrennt adressiert, indem der Sequencer und der Grafik-Controller programmiert werden.

Richard Wilton

Dieser Text stammt aus dem Buch Buch »The programmer's guide to PC and PS/2 video systems. Maximum performance from the EGA, VGA, HGC, and MCGA« von Richard Wilton, erschienen bei Microsoft Press.

Die englische Ausgabe haben wir im Microsoft System Journal Juli/August 1988 auf der Seite 47 besprochen. Die deutsche Übersetzung wird in Kürze unter dem Titel »Text- und Grafikdarstellung mit IBM PC und IBM PS/2« (ISBN 3-528-04627-9, Preis ca. DM 90,-) im Vieweg-Verlag erscheinen, mit dessen freundlicher Genehmigung der Abdruck erfolgt.

Wir möchten dieses Buch unseren Lesern an dieser Stelle noch einmal ausdrücklich empfehlen. Wer gute und schnelle Bildschirmausgaberoutinen in Assembler oder für C-Programme benötigt, sollte sich dieses Buch (am besten gleich mit Diskette) zulegen.

```
                TITLE   'Listing 1'
                NAME    ScreenDumpCGA
                PAGE    55,132

;
; Name:         ScreenDumpCGA
;
; Funktion:     Bildschirmausdruck für
;               CGA 640x200 2-farbig und 320x200 4-farbig
;
; Aufrufendes Programm: (undefiniert)
;
; Hinweis:      Die Hauptprozedur dieses Programms,
;               ScreenDumpCGA, kann aus jedem Anwendungs-
;               programm oder als Teil eines TSR-
;               Handlers (Terminate-but-Stay Resident)
;               für Interrupt 5 aufgerufen werden.
;

STDPRN  =       4                       ; MS-DOS, Standarddruckernummer


DGROUP  GROUP   _DATA

_TEXT   SEGMENT byte public 'CODE'
        ASSUME  cs:_TEXT,ds:DGROUP

;
; PrintLine
;
;       Schreibt eine Zeile in die Standarddruckereinheit.
;       Fehler werden ignoriert.
;

PrintLine PROC  near                    ; Aufr. Pgm.: DS:DX -> Daten
                                        ;             CX = Anzahl Bytes
        mov     bx,STDPRN
        mov     ah,40h                  ; INT 21h Funk.40h: Schreiben
        int     21h
        ret

PrintLine ENDP
```

```
;
; PrinterGraphics
;
;       Versetzt Drucker in den "Grafikmodus".  Diese Routine
;       muß für verschiedene Drucker angepaßt werden.
;

PrinterGraphics PROC near       ; Konfiguriert Epson MX-80
                                ;  auf 480 Punkte/Zeile

        mov     dx,offset DGROUP:EpsonGraphics
        mov     cx,3
        call    PrintLine
        ret

PrinterGraphics ENDP

;
; PrinterDefault
;
;       Versetzt Drucker in den Standardmodus (keine Graphik).
;       Diese Routine muß ebenfalls angepaßt werden.
;

PrinterDefault PROC     near    ; Konfiguriert Epson MX-80 für
                                ;  alphanumerische Ausgabe
                                ;  (Standard)

        mov     dx,offset DGROUP:EpsonReset
        mov     cx,2
        call    PrintLine
        ret

PrinterDefault  ENDP

;
; ChopZeroes
;
;       Schneidet Nullen am Ende der Druckerausgabe (Puffer) ab
;

ChopZeroes PROC near            ; Aufr. Pgm.:ES:DI -> Puffer
                                ;            CX=Pufferlänge
                                ; Rückgabe:  CX=angepaßte Länge

        jcxz    L01             ; Aussprung, wenn Puffer leer

        add     di,cx
        dec     di              ; ES:DI->letztes Byte im Puffer

        xor     al,al           ; AL := 0 (zu suchendes Byte)

        std                     ; Rückwärtssuche
        repe    scasb
        cld                     ; Direction-Flag wiederherst.
        je      L01             ; Sprung, wenn Puffer mit
                                ;  Nullen gefüllt

        inc     cx              ; Länge auf letztes Byte
                                ;  ungleich 0 setzen

L01:    ret

ChopZeroes ENDP

;
; PrintPixels
;
;       Druckt eine Zeile von Pixeln auf einem Epson MX-80.
;

PrintPixels PROC near           ; Aufr. Pgm.: DI = Pufferoffset
                                ;             CX = Pufferlänge

        push    ds
        pop     es              ; ES := DS

        push    di              ; Pufferoffset sichern
        call    ChopZeroes
        push    cx              ; Länge sichern
        mov     word ptr DataHeader+2,cx  ; Pufferlänge im Aus-
                                ;  gabedaten-Header speichern
        mov     dx,offset DGROUP:DataHeader
        mov     cx,4
        call    PrintLine       ; Daten-Header drucken

        pop     cx              ; CX := Pufferlänge
        pop     dx              ; DX := Pufferoffset
        call    PrintLine       ; Pixel drucken

        mov     dx,offset DGROUP:CRLF
        mov     cx,2
        call    PrintLine

        ret

PrintPixels ENDP

;
; TranslatePixels
;
;       Kopiert eine druckbare Pixelzeile aus dem Bildschirm-
;       puffer in den Druckerpuffer. Diese Routine kann
;       verändert werden, um die Skalierung oder Ausrichtung
;       eines Abdruckes anzupassen, Farbdarstellungen in
;       Graustufen umzuwandeln usw.
;
;       Diese Routine formatiert den Druckerpuffer für die
;       Ausgabe auf einem Epson MX-80. Die Seite wird um 90
;       Grad gedreht ausgedruckt, wobei zwei horizontal
;       gedruckte Pixel für jedes vertikale Pixel im Bild-
;       schirmpuffer erscheinen. Da der CGA-Bildschirm eine
;       Höhe von 200 Pixeln besitzt, resultiert ein Ausdruck
;       von 400 Pixel Breite.
;

TranslatePixels PROC near       ; Aufr. Pgm.:
                                ; SI = Bildschirmpufferoffset
                                ; ES:DI->Druckerpuffer

        push    ds              ; DS sichern
        mov     ds,VideoBufSeg  ; DS:SI->Bildschirmpuffer

        add     di,398          ; ES:DI->2 Bytes vor Pufferende

        mov     cx,200          ; CX := # vertikaler Pixel
        mov     bx,2000h+1      ; BX := 1. Bildschirmpufferink.
        mov     dx,81-2000h     ; DX := 2. Bildschirmpufferink.

        std                     ; Druckpuffer rückwärts füllen

L11:    lodsb                   ; AL := 8 Pixel des Bildschirm-
                                ;  puffers
        mov     ah,al           ; AX := 8 doppelte Pixel
        stosw                   ; in Druckerpuffer schreiben

        add     si,bx           ; Inkrementieren zum nächsten
                                ;  Interleave
        xchg    bx,dx           ; des Bildschirmpuffers

        loop    L11

        cld                     ; Direction-Flag löschen
        pop     ds              ; DS wiederherstellen
        ret

TranslatePixels ENDP

;
; ScreenDumpCGA
;

ScreenDumpCGA PROC near         ; Aufr. Pgm.:   DS = DGROUP

        call    PrinterGraphics ; Konfigurieren des Druckers:
                                ;  Grafik

        push    ds
        pop     es              ; DS,ES := DGROUP
```

```
        xor     si,si             ; SI := Startoffset des
                                  ;  Bildschirmpuffers

L21:    push    si
        mov     di,offset DGROUP:PrintBuf
        call    TranslatePixels ; Kopieren einer druckbaren
                                ;  Pixelzeile

        mov     cx,400
        mov     di,offset DGROUP:PrintBuf
        call    PrintPixels     ; drucken

        pop     si
        inc     si
        cmp     si,80           ; Schleife über alle 80 Spalten
        jb      L21             ;  im Bildschirmpuffer

        call    PrinterDefault  ; Druckerstandardmodus
                                ;  wiederherstellen
                ret

ScreenDumpCGA ENDP

_TEXT   ENDS

_DATA           SEGMENT word public 'DATA'

PrintBuf        DB      400 dup(?)      ; Ausgabepuffer drucken

VideoBufSeg     DW      0B800h

EpsonGraphics   DB      1Bh,33h,18h
EpsonReset      DB      1Bh,40h
DataHeader      DB      1Bh,4Bh,00h,00h
CRLF            DB      0Dh,0Ah

_DATA           ENDS

                END
```

Listing 1: *Einfache Bildschirmausdruckroutine für CGA*

```
                TITLE   'Listing 2'
                NAME    ScreenDumpEGA
                PAGE    55,132

;
; Name:         ScreenDumpEGA
;
; Funktion:     Bildschirmausdruck für EGA 640x350 16-farbig
;
; Aufrufendes Pgm.:     (undefiniert)
;
; Hinweis:      Die Hauptprozedur dieses Programms,
;               ScreenDumpEGA, kann aus jedem Anwendungs-
;               programm oder als Teil eines TSR-
;               Handlers (Terminate-but-Stay Resident)
;               für Interrupt 5 aufgerufen werden.
;

STDPRN  =       4               ; MS-DOS, Standarddruckernummer

DGROUP  GROUP   _DATA

_TEXT   SEGMENT byte public 'CODE'
        ASSUME  cs:_TEXT,ds:DGROUP
```

```
;
; PrintLine
;
;       Schreibt eine Zeile mit Zeichen zur Standarddrucker-
;       einheit. Fehler werden ignoriert.
;

PrintLine PROC  near            ; Aufr. Pgm.: DS:DX -> Daten
                                ;             CX = Anzahl Bytes
        mov     bx,STDPRN
        mov     ah,40h          ; INT 21h Funk.40h:  Schreiben
        int     21h
        ret

PrintLine ENDP

;
; PrinterGraphics
;
;       Versetzt Drucker in den "Grafikmodus".  Diese Routine
;       muß für verschiedene Drucker angepaßt werden.
;

PrinterGraphics PROC near       ; Konfiguriert Epson MX-80
                                ;  auf 480 Punkte/Zeile

        mov     dx,offset DGROUP:EpsonGraphics
        mov     cx,3
        call    PrintLine
        ret

PrinterGraphics ENDP

;
; PrinterDefault
;
;       Versetzt Drucker in den Standardmodus (keine Grafik).
;       Diese Routine muß für verschiedene Drucker angepaßt
;       werden.
;

PrinterDefault  PROC    near    ; Konfiguriert Epson MX-80 im
                                ; Standardmodus(alphanumerisch)

        mov     dx,offset DGROUP:EpsonReset
        mov     cx,2
        call    PrintLine
        ret

PrinterDefault  ENDP

;
; ChopZeroes
;
;       Schneidet Nullen am Ende des Druckerausgabepuffers ab.
;

ChopZeroes PROC near            ; Aufr. Pgm.: ES:DI -> Puffer
                                ;             CX = Pufferlänge
                                ; Rückgabe:
                                ;         CX = angepaßte Länge
        jcxz    L01             ; Aussprung, wenn Puffer leer

        add     di,cx
        dec     di              ; ES:DI->letztes Byte im Puffer

        xor     al,al           ; AL := 0 (zu suchendes Byte)

        std                     ; Rückwärtssuche
        repe    scasb
        cld                     ; Direction-Flag wiederherst.
        je      L01             ; Sprung, wenn Puffer mit
                                ;  Nullen gefüllt

        inc     cx              ; Länge auf letztes Byte
                                ; ungleich Null anpassen

L01:    ret

ChopZeroes ENDP
```

```
;
; PrintPixels
;
;       Druckt eine Pixelzeile auf einem Epson MX-80.
;

PrintPixels PROC near                 ; Aufr. Pgm.: DI = Pufferoffset
                                      ;             CX = Pufferlänge

        push    ds
        pop     es                    ; ES := DS

        push    di                    ; Pufferoffset sichern
        call    ChopZeroes
        push    cx                    ; Länge sichern

        mov     word ptr DataHeader+2,cx  ; Pufferlänge im Aus-
                                      ; gabedaten-Header speichern
        mov     dx,offset DGROUP:DataHeader
        mov     cx,4
        call    PrintLine             ; Daten-Header drucken

        pop     cx                    ; CX := Pufferlänge
        pop     dx                    ; DX := Pufferoffset
        call    PrintLine             ; Pixel drucken

        mov     dx,offset DGROUP:CRLF
        mov     cx,2
        call    PrintLine

        ret

PrintPixels ENDP


;
; TranslatePixels
;
;       Kopiert eine druckbare Pixelzeile aus dem Bildschirm-
;       puffer in den Druckerpuffer. Diese Routine kann
;       verändert werden, um die Skalierung oder Ausrichtung
;       eines Abdruckes anzupassen, Farbdarstellungen in
;       Graustufen umzuwandeln usw.
;
;       Diese Routine formatiert den Druckerpuffer für die
;       Ausgabe auf einem Epson MX-80. Die Seite wird um 90
;       Grad gedreht ausgedruckt. Es resultiert ein Ausdruck
;       von 350 Pixeln Breite.
;

TranslatePixels PROC near             ; Aufr. Pgm.:
                                      ; SI = Bildschirmpufferoffset
                                      ; ES:DI -> Druckerpuffer

        push    ds                    ; DS sichern
        mov     ds,VideoBufSeg        ; DS:SI -> Bildschirmpuffer

        add     di,349                ; ES:DI -> letzes Byte im
                                      ;  Druckerpuffer

        mov     cx,350                ; CX := Anzahl vertikaler Pixel

; Initialisierung des Grafik-Controllers auf Lesemodus 1

        mov     dx,3CEh               ; Grafik-Controller E/A-Port
        mov     ax,805h               ; AH := 00001000b (Lesemodus 1)
                                      ; AL := Modusregisternummer
        out     dx,ax

        mov     ax,002                ; AH := 0 (Farbvergleichswert)
        out     dx,ax                 ; AL := Color-Compare-Register-
                                      ;  Nummer

        mov     ax,0F07h              ; AH := 00001111b
                                      ;  (Color-Don't-Care-Maske)
        out     dx,ax                 ; AL := Color-Don't-Care-
                                      ;  Registernummer

; Druckerpuffer füllen; alle Pixel <> 0 im Bildschirmpuffer
; werden gedruckt

        std                           ; Druckerpuffer rückwärts füllen

L11:    lodsb                         ; AL := 8-Pixel-Farbver-
                                      ;  gleichswert
                                      ;  (Bits = 0, wenn Pixel <> 0)
        not     al                    ; AL := 8 druckbare Pixel
        stosb                         ; im Druckerpuffer speichern

        add     si,81                 ; inkrementieren: nächste
                                      ; Bildschirmpufferzeile
        loop    L11

        cld                           ; Direction-Flag löschen

; Grafik-Controller-Standardstatus wiederherstellen

        mov     ax,5                  ; AH := Lesemodus 0,
                                      ;  Schreibmodus 0
        out     dx,ax                 ; AL := Modusregisternummer
        mov     ax,7                  ; AH := 0 (Color-Don't-Care-
                                      ;  Maske)
        out     dx,ax                 ; AL := Color-Don't-Care-
                                      ;  Registernummer

        pop     ds                    ; DS wiederherstellen
        ret

TranslatePixels ENDP

;
; ScreenDumpEGA
;

ScreenDumpEGA PROC near               ; Aufr. Pgm.:    DS = DGROUP
        call    PrinterGraphics       ; Konfigurieren des Drucker
                                      ;  auf Grafik

        push    ds
        pop     es                    ; DS,ES := DGROUP

        xor     si,si                 ; SI := Startoffset des
                                      ;  Bildschirmpuffers

L21:    push    si
        mov     di,offset DGROUP:PrintBuf
        call    TranslatePixels       ; Kopieren einer druckbaren
                                      ; Pixelzeile
        mov     cx,350
        mov     di,offset DGROUP:PrintBuf
        call    PrintPixels           ; ausgeben

        pop     si
        inc     si
        cmp     si,80                 ; Schleife über alle 80 Spalten
        jb      L21                   ;  im Bildschirmpuffer

        call    PrinterDefault        ; Standarddruckerstatus
                                      ; wiederherstellen
        ret
ScreenDumpEGA   ENDP

_TEXT   ENDS

_DATA           SEGMENT word public 'DATA'
PrintBuf        DB      350 dup(?)      ; Ausgabepuffer

VideoBufSeg     DW      0A000h

EpsonGraphics   DB      1Bh,33h,18h
EpsonReset      DB      1Bh,40h
DataHeader      DB      1Bh,4Bh,00h,00h
CRLF            DB      0Dh,0Ah
_DATA           ENDS

                END
```

Listing 2: *Routine für einen EGA-Bildschirmausdruck*

```
                TITLE   'Listing 3'
                NAME    ScreenDumpAlpha
                PAGE    55,132

;
; Name:         ScreenDumpAlpha
;
; Funktion:     Bildschirmausdruck für EGA (alphanumerische
;               Modi mit 350-zeiliger Auflösung
;
; Aufr. Pgm.:   (undefiniert)
;
; Hinweis:      Die Hauptprozedur des Programms,
;               ScreenDumpAlpha, kann aus einem Anwendungs-
;               programm oder als Teil eines TSR-Handlers
;               (Terminate-but-Stay Resident) für
;               Interrupt 5 aufgerufen werden.
;

STDPRN  =       4               ; MS-DOS, Standarddruckernummer

DGROUP  GROUP   _DATA

_TEXT   SEGMENT byte public 'CODE'
        ASSUME  cs:_TEXT,ds:DGROUP,es:DGROUP


;
; PrintLine
;
;       Schreibe eine Zeile mit Zeichen zur
;       Standarddruckereinheit. Fehler werden ignoriert.
;

PrintLine PROC  near            ; Aufr. Pgm.: DS:DX -> Daten
                                ;             CX = Anzahl Bytes
        mov     bx,STDPRN
        mov     ah,40h          ; INT 21h Funk.40h:  Schreiben
        int     21h
        ret

PrintLine ENDP


;
; PrinterGraphics
;
;       Versetzt Drucker in den "Grafikmodus".  Diese Routine
;       muß für verschiedene Drucker angepaßt werden.
;

PrinterGraphics PROC near       ; Konfiguriert Epson MX-80
                                ;  auf 480 Punkte/Zeile

        mov     dx,offset DGROUP:EpsonGraphics
        mov     cx,3
        call    PrintLine
        ret

PrinterGraphics ENDP


;
; PrinterDefault
;
;       Versetzt Drucker in den Standardmodus (keine Grafik).
;       Diese Routine muß für verschiedene Drucker angepaßt
;       werden.
;

PrinterDefault PROC near        ; Konfiguriert Epson MX-80 auf
                                ;  alphanumerische Ausgabe

        mov     dx,offset DGROUP:EpsonReset
        mov     cx,2
        call    PrintLine
        ret

PrinterDefault ENDP

;
; ChopZeroes
;
;       Nullen am Ende des Druckerausgabepuffers werden
;       abgeschniten.
;

ChopZeroes PROC near            ; Aufr. Pgm.: ES:DI -> Puffer
                                ;             CX = Pufferlänge
                                ; Returns: CX = angepaßte Länge
        jcxz    L01             ; Aussprung, wenn Puffer leer

        add     di,cx
        dec     di              ; ES:DI->letztes Byte im Puffer

        xor     al,al           ; AL := 0 (zu suchendes Byte)

        std                     ; Rückwärtssuche
        repe    scasb
        cld                     ; Direction-Flagge wiederherst.
        je      L01             ; Sprung, wenn Puffer mit
                                ;  Nullen gefüllt
        inc     cx              ; Länge auf letztes Byte
                                ;  <> Null setzen

L01:    ret

ChopZeroes ENDP


;
; PrintPixels
;
;       Druckt eine Pixelzeile auf einem Epson MX-80.
;


PrintPixels PROC near           ; Aufr. Pgm.: DI = Pufferoffset
                                ;             CX = Pufferlänge

        push    ds
        pop     es              ; ES := DS

        push    di              ; Pufferoffset sichern
        call    ChopZeroes
        push    cx              ; Länge sichern

        mov     word ptr DataHeader+2,cx  ; Pufferlänge im
                                ; Ausgabedaten-Header speichern
        mov     dx,offset DGROUP:DataHeader
        mov     cx,4
        call    PrintLine       ; Daten-Header drucken

        pop     cx              ; CX := Pufferlänge
        pop     dx              ; DX := Pufferoffset
        call    PrintLine       ; Pixel drucken

        mov     dx,offset DGROUP:CRLF
        mov     cx,2
        call    PrintLine

        ret

PrintPixels ENDP


;
; TranslatePixels
;
;       Kopiert eine druckbare Pixelzeile aus der ersten
;       Zeichendefinitionstabelle (Map 2) in den Druckerpuffer.
;
;       Diese Routine formatiert den Druckerpuffer für die
;       Ausgabe auf einem Epson MX-80. Die Seite wird um 90
;       Grad gedreht ausgedruckt. Es resultiert ein Ausdruck
;       von 350 Pixeln Breite.
;
```

```
TranslatePixels PROC near           ; Aufr. Pgm.:
                                    ;  SI = Bildschirmpufferoffset
                                    ;  ES:DI -> Puffer drucken

        push    ds                  ; DS sichern
        mov     ds,VideoBufSeg      ; DS:SI -> Bildschirmpuffer

        add     di,es:PrintBufSize
        dec     di                  ; ES:DI -> letztes Byte im
                                    ;  Druckerpuffer

        mov     dx,3CEh             ; Grafik-Controller E/A-Port

; Druckerpuffer füllen

        mov     cx,es:Rows          ; CX := Anzahl Zeichenzeilen

L11:    push    cx                  ; CX und SI sichern
        push    si

        mov     ax,0004h            ; AH := Wert für Read-Map-
                                    ;  Select-Register
                                    ; AL := Read-Map-Select-
                                    ;  Registernummer
        out     dx,ax               ; Auswahl von Map 0
                                    ;  (Zeichencodes)

        lodsb                       ; AX := nächster Zeichencode
                                    ;  im Bildschirmpuffer
        mov     cl,5
        shl     ax,cl               ; AX := AX * 32
        mov     si,ax               ; SI := Offset der Zeichen-
                                    ;  definition in Map 2
        mov     ax,0204h
        out     dx,ax               ; Auswahl von Map 2 (Bitmuster)

        mov     cx,es:Points        ; CX := Größe der Zeichen-
                                    ;  definition

L12:    cld
        lodsb                       ; AL := 8-Bit-Muster der
                                    ; Zeichendefinitionstabelle
                                    ; SI := SI + 1
        std
        stosb                       ; Bitmuster im Druckerpuffer
                                    ;  speichern, DI := DI - 1
        loop    L12                 ; Schleife durch die Zeichen-
                                    ;  definition

        pop     si                  ; SI und CX wiederherstellen
        pop     cx

        add     si,es:Columns       ; DS:SI -> nächste Zeichenzeile
        loop    L11                 ; Schleife durch Zeichenzeilen

        cld                         ; Direction-Flag löschen

        pop     ds                  ; DS wiederherstellen
        ret
TranslatePixels ENDP

;
; ScreenDumpAlpha
;

ScreenDumpAlpha PROC near           ; Aufr. Pgm.:   DS = DGROUP

        call    PrinterGraphics     ; Konfigurieren der Grafik-
                                    ;  druckerausgabe

        call    CGenModeSet         ; EGA-Speicher-Maps parallel:
                                    ;  Map 0 enthält Zeichencodes
                                    ;  Map 2 enthält Zeichen-
                                    ;  definitionen

; Bildschirmdimensionen vom Bildschirmdatenbereich kopieren

        mov     ax,40h
        mov     es,ax               ; ES -> BIOS-Bildschirmdaten-
                                    ;  bereich
        mov     al,es:[84h]         ; AX := ROWS
        inc     ax
        mov     Rows,ax
        mov     ax,es:[4Ah]         ; AX := CRT_COLS
        add     ax,ax               ; * 2 zur korrekten Puffer-
                                    ;  adressierung
        mov     Columns,ax
        mov     ax,es:[85h]         ; AX := POINTS
        mov     Points,ax
        mul     Rows                ; AX := ROWS * POINTS
        mov     PrintBufSize,ax

; Bildschirm drucken
        push    ds
        pop     es                  ; DS,ES := DGROUP

        xor     si,si               ; SI := Startoffset des
                                    ;  Bildschirmpuffers

L21:    push    si
        mov     di,offset DGROUP:PrintBuf
        call    TranslatePixels     ; eine druckbare Pixelzeile
                                    ;  kopieren

        mov     cx,PrintBufSize
        mov     di,offset DGROUP:PrintBuf
        call    PrintPixels         ; drucken

        pop     si
        add     si,2                ; Inkrementieren auf nächste
                                    ;  Spalte
        cmp     si,Columns          ; Schleife über alle Zeichen-
                                    ;  spalten
        jb      L21

        call    CGenModeClear       ; vorherigen Textmodus wieder-
                                    ;  herstellen
        call    PrinterDefault      ; Standarddruckerstatus
                                    ;  wiederherstellen
        ret
ScreenDumpAlpha ENDP

;
; CGenModeSet
;

CGenModeSet PROC near

        push    si                  ; Register sichern
        push    cx

        cli                         ; Interrupts unterbinden
        mov     dx,3C4h             ; Sequencer-Portadresse
        mov     si,offset DGROUP:SetSeqParms
        mov     cx,4

L31:    lodsw                       ; AH := Wert für Sequencer-
                                    ;  Register
                                    ; AL := Registernummer
        out     dx,ax               ; Programmieren des Registers
        loop    L31
        sti                         ; Interrupts wieder ermöglichen

        mov     dl,0CEh             ; DX := 3CEH (Grafik-Con-
                                    ;  troller-Port-Adresse)
        mov     si,offset DGROUP:SetGCParms
        mov     cx,3

L32:    lodsw                       ; Grafik-Controller pro-
                                    ;  grammieren
        out     dx,ax
        loop    L32

        pop     cx                  ; Register wiederherstellen
        pop     si                  ; Rücksprung
        ret
CGenModeSet ENDP

; CGenModeClear
;
```

```
CGenModeClear PROC near

        push    si                      ; Register sichern
        push    cx

        cli                             ; Interrupts unterbinden
        mov     dx,3C4h                 ; Sequencer-Portadresse
        mov     si,offset DGROUP:ClearSeqParms
        mov     cx,4

L41:    lodsw                           ; AH := Wert für Sequencer-
                                        ;  Register
                                        ; AL := Registernummer
        out     dx,ax                   ; Programmieren des Registers
        loop    L41
        sti                             ; Interrupts wieder ermöglichen

        mov     dl,0CEh                 ; DX := 3CEH (Grafik-Con-
                                        ;  troller-Portadresse)
        mov     si,offset DGROUP:ClearGCParms
        mov     cx,3

L42:    lodsw                           ; Programmieren des Grafik-
                                        ;  Controllers
        out     dx,ax
        loop    L42

        mov     ah,0Fh                  ; AH := INT 10H Funktionsnummer
        int     10h                     ; Bildschirmmodus holgen

        cmp     al,7
        jne     L43                     ; Sprung, wenn kein Monochrom-
                                        ;  modus
        mov     ax,0806h                ; Grafik-Controller pro-
                                        ;  grammieren für Map-Beginn
        out     dx,ax                   ;  bei B000:0000

L43:    pop     cx                      ; Register wiederherstellen
        pop     si                      ; Rücksprung
        ret
CGenModeClear ENDP

_TEXT   ENDS


_DATA           SEGMENT word public 'DATA'
PrintBuf        DB      400 dup(?)      ; Ausgabepuffer

VideoBufSeg     DW      0A000h

EpsonGraphics   DB      1Bh,33h,18h
EpsonReset      DB      1Bh,40h
DataHeader      DB      1Bh,4Bh,00h,00h
CRLF            DB      0Dh,0Ah

Columns         DW      ?               ; Anzahl Zeichenspalten
Rows            DW      ?               ; Anzahl Zeichenzeilen
Points          DW      ?               ; Vertikale Größe Zeichenmatrix
PrintBufSize    DW      ?               ; Zeilen * Punkte

SetSeqParms     DW      0100h           ; Parameter für CGenModeSet
                DW      0402h
                DW      0704h
                DW      0300h
SetGCParms      DW      0204h
                DW      0005h
                DW      0006h
ClearSeqParms   DW      0100h           ; Parameter für CGenModeClear
                DW      0302h
                DW      0304h
                DW      0300h
ClearGCParms    DW      0004h
                DW      1005h
                DW      0E06h
_DATA           ENDS
                END
```

Listing 3: *Verwendung RAM-gestützter Zeichendefinitionstabellen zum Ausdruck des Zeichensatz*

C für Mikroprozessor-Entwicklungsprojekte:

ROM-fähige Programme mit Microsoft C

Software ist der Schlüssel zum Erfolg von Mikroprozessor-Produkten. Die Zunahme der Komplexität der heutigen 16-Bit- und 32-Bit-Mikroprozessoren zusammen mit den ansteigenden Entwicklungskosten und dem Druck, die Produkte auf den Markt zu bringen, diktieren die Anwendung von Hochsprachen zur Erhöhung der Produktivität. C ist zur Hochsprache der Wahl für Mikroprozessor-Entwicklungsprojekte geworden.

C wird üblicherweise in einer RAM-residenten Hostumgebung mit einer umfangreichen Run-Time-Umgebung (z.B. UNIX oder MS-DOS) angewendet. Das Programm kann schnell und einfach getestet werden, indem das Programm direkt auf dem System, auf dem es geschrieben wurde, ausgeführt wird. Dies gilt jedoch nicht für Systeme mit eingebetteten Mikroprozessoren, wie Maschinensteuerungen oder computerisierte Geräte. Diese Applikationen sind meist eigenständige Systeme, das heißt, Systeme mit eingebetteten Mikroprozessoren haben weder ein Betriebssystem, noch eine Umgebung zur Unterstützung einer Benutzerschnittstelle oder eines Dateisystems. Daher stellt der Einsatz von C bei der Mikroprozessor-Systementwicklung für eingebettete Anwendungen eine neue Herausforderung dar. Dazu gehört auch ROM-Unterstützung, Programmierbarkeit auf der Zielhardware und Debug-Support.

Bei der Auswahl eines C-Compilers für die Unterstützung der Programmentwicklung für eine eingebettete Anwendung sollte der Designer eine Umgebung mit folgenden Eigenschaften wählen: optimale Codegenerierung, Software-Integration für das Zielsystem und Debugging-Fähigkeit - alles in einem integrierten Paket. Der Microsoft C Optimizing Compiler und Makro-Assemblerbietet zusammen mit dem SSI Link & Locate Firmware Generierungspaket für Microsoft C und dem SoftProbe II/TX Cross Debugger eine solche Lösung.

Das Microsoft Compiler-System entspricht dem Entwurf des ANSI-C-Standards und kann Code für den 8086-, 80186- oder 80286-Mikroprozessor generieren. Das SSI Link & Locate-Paket unterstützt nur die Befehle des 8086 und 80186. Microsoft C unterstützt zwei wichtige Eigenschaften: Register-Variable zur Optimierung der Ausführungsgeschwindigkeit, die für eingebettete Echtzeit-Systemanwendungen notwendig ist; und das Schlüsselwort `const` wird vom Compiler als Typ-Modifizierer für bestimmte Objekte unterstützt, die read-only sind.

Damit soll jede Zuordnung zum Objekt oder andere Nebeneffekte wie Increment oder Decrement verhindert werden. Der Compiler generiert eine Fehlermeldung, wenn `const`-Objekte modifiziert werden. Der Microsoft C-Compiler generiert optimierten reentrant-fähigen Code, der bei der Entwicklung von eingebetteten Echtzeit Mikroprozessoranwendungen eingesetzt werden kann.

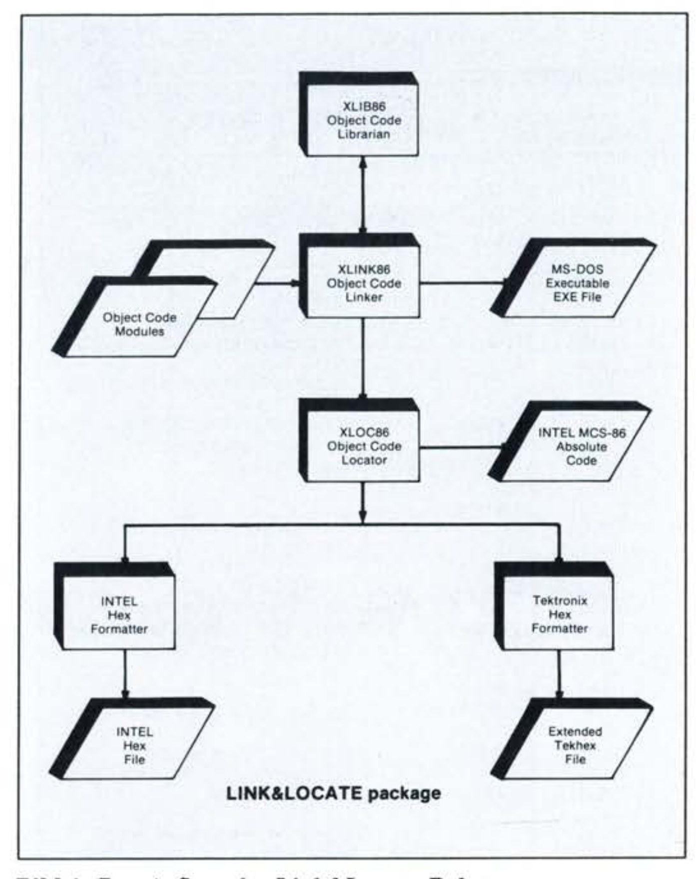


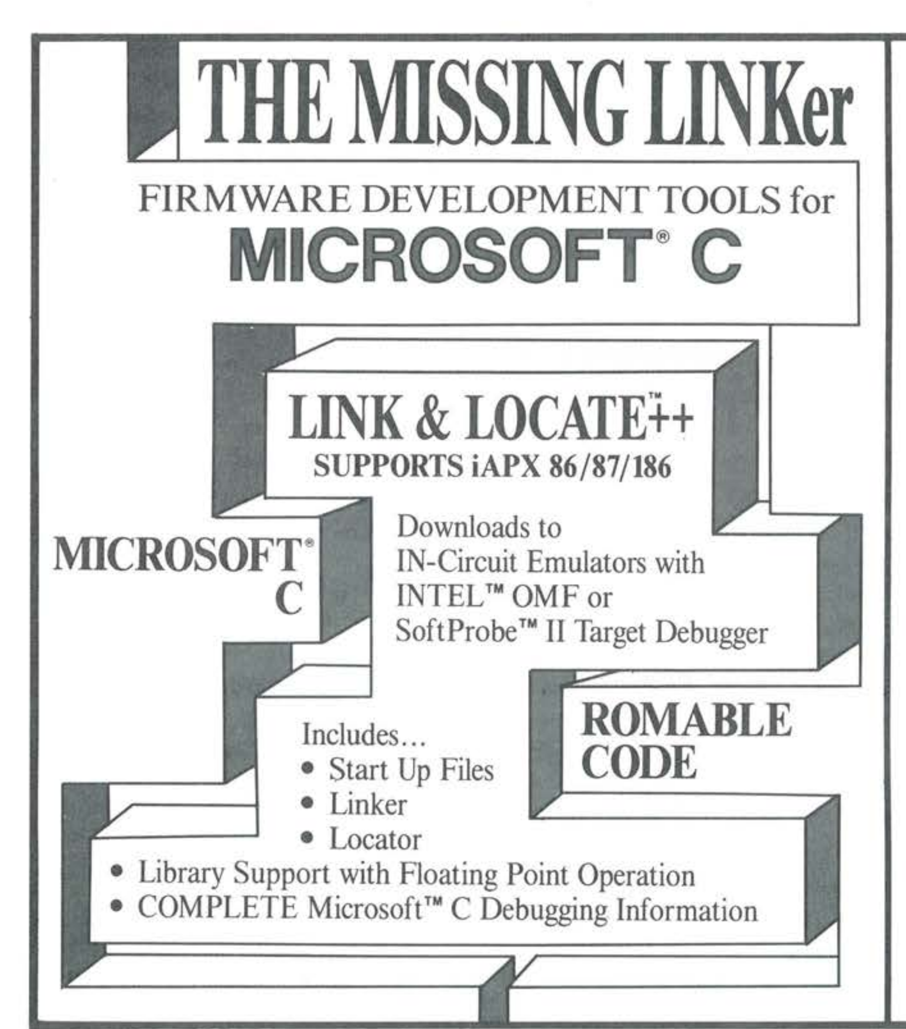
Bild 1: *Der Aufbau des Link&Locate-Pakets.*

Das Microsoft C-Compiler-Paket enthält ein Runtime Library-Paket zur Unterstützung der MS-DOS Betriebssystem-Umgebung. Die MS-DOS-Funktionen machen Systemaufrufe zu dem darunterliegenden Betriebssystem. Somit können sie nicht für eingebettete Systemanwendungen eingesetzt werden.

Die MS-DOS-unabhängigen Funktionen benötigen keine Betriebssystemunterstützung und sie rufen keine MS-DOS- oder BIOS-abhängigen Funktionen auf. Diese Gruppe von Funktionen kann bei eingebetteten Systemanwendungen eingesetzt werden.

Die Runtime Library, die im Microsoft C-Compiler enthalten ist, kann mit dem SSI XLINK86 Linker nicht ohne Modifikation gebunden werden. Anwender sollten von dem Programm LIB Gebrauch machen, um die von MS-DOS unabhängigen Funktionen in separate Objektdateien zu extrahieren und sie dann mit XLINK86 in eine Librarydatei zu binden. Eine andere Methode ist die Anwendung des CNVLIB Formatierers, um Microsoft Runtime-Bibliotheksdateien direkt in Intel OMF-Dateien umzuwandeln. Das CNVLIB Programm ist im Link & Locate Paket enthalten.

In der modularen Programmierung wird ein Programm in viele Module aufgeteilt. Die Objektdateien und gegebenenfalls die Librarydateien werden dann später zu einer einzigen relokativen Objektdatei zusammengefaßt.


THE MISSING LINKer
FIRMWARE DEVELOPMENT TOOLS for
MICROSOFT® C
LINK & LOCATE™++
SUPPORTS iAPX 86/87/186
MICROSOFT® C
Downloads to
IN-Circuit Emulators with
INTEL™ OMF or
SoftProbe™ II Target Debugger
ROMABLE CODE
Includes...
• Start Up Files
• Linker
• Locator
• Library Support with Floating Point Operation
• COMPLETE Microsoft™ C Debugging Information

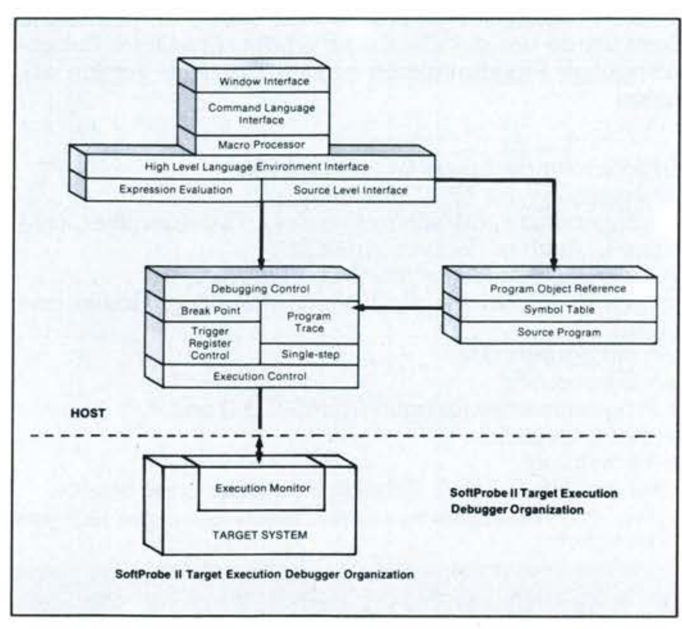


Bild 2: *Der Aufbau des SoftProbe II/TX Cross Debuggers.*

Mit dem SSI XLINK86 Linker lassen sich die Objektdateien zu einer einzigen relokativen Objektdatei binden. Eine Link-Map liefert eine Aufstellung aller logischen Segmente, die in der relokatierbaren Objektdatei enthalten sind.

Der SSI XLOC86 Locator ordnet die absoluten Adressen aller logischen Segmente in einer relokatierbaren Datei zu. Alle Segmente in einem Programm können überall in dem 1 Mbyte Adreßbereich des iAPX-86 Mikroprozessors angeordnet werden. Adreßzuweisung kann in einzelnen Segmenten wie auch in Segmentgruppen erfolgen.

Der Locator erstellt eine absolute Objektdatei im Intel-OMF-Format und ein Locate-Verzeichnis. Die Locate-Map liefert eine Zusammenfassung der absoluten Adressen in allen Segmenten der absoluten Objektdatei. Die absoluten Werte aller globalen Symbole sind ebenfalls enthalten.

Weitere Dienstprogramme sind in dem Link & Locate-Paket enthalten: ein PROM86-Formatierer, der zur Bereitung eines absoluten Objektfiles in eine oder mehrere Dateien für ein EPROM Programmiergerät dient; ein XOH86-Formatierer zur Umwandlung eines absoluten Objektfiles in eine Intel 8086 Hexdatei, und ein TEKHEX-Formatierer zur Umwandlung einer absoluten Objektdatei in ein Tektronix Extended Hexfile.

Software-Debugging auf einem Zielsystem

Debugging von eingebetteten Systemanwendungen, die in der C Sprache geschrieben wurden, ist sehr verschieden von dem Debugging eines Programms, das für MS-DOS-Anwendungen geschrieben wurde. Dies führt darauf zurück, daß die Zielsysteme meist selbständig sind und weder ein Betriebssystem noch eine Benutzerschnittstelle haben, um das Debugging zu erleichtern. Ein Hilfsmittel, eingebettete Systemanwendungen zu testen, ist der SoftProbe II/TX Cross Debugger. Er ist ein interaktiver Software Debugger und erlaubt dem Benutzer, Anwendungsprogramme von einem PC Hostcomputer in das Zielsystem herunterzuladen und die Programme dort auszuführen. Die Anwendungsprogramme können auf C-Ebene oder auf Assemblerebene ausgetestet werden.

Das SoftProbeII/TX Paket enthält ein Benutzer-Interface-Programm für den Host-Computer und ein Monitorprogramm für das Zielsystem. Das Monitorprogramm wird im Quellcode geliefert, so daß die Anwender es für ihre Zielsysteme anpassen können. Das angepaßte Programm sollte in ein EPROM eingebrannt und in das Zielsysten eingesteckt werden. Das Monitorprogramm hat 4 Kbyte Code und benötigt 2 Kbyte Datenspeicher.

Das Zielsystem ist an den PC über eine V-24 Schnittstelle mit einer Übertragungsrate von 9600 Baud verbunden. Das Benutzer-Interface-Programm auf dem Hostcomputer dient dazu, den absoluten Code vom Hostsystem in den Speicher des Zielsystems zu laden. Symbolinformation bleibt auf dem Hostsystem erhalten, um symbolisches Debuggen zu erleichtern. Das Anwendungsprogramm kann im Zielsystem sowohl auf der C-Ebene als auch auf Assemblerebene getestet werden.

Der Microsoft C-Compiler kann Debuginformation im Intel-OMF-Format erzeugen. Über den Schalter /Zd werden Debug-Informationen wie Zeilennummern und globale Sybolnamen in die Objektdatei übernommen. Objektdateien, die die Debug-Informationen enthalten, werden von dem Link & Locate-Paket verarbeitet, um absolute Objektdateien im Intel-OMF-Format zu erzeugen. Diese Debug-Informationen in den absoluten Objektdateien ermöglichen dem Anwender von SoftProbe II/TX unter anderem folgendes:

- Bezugnahme auf anwenderdefinierte und Original-Programmvariablen durch Symbolnamen
- Bezugnahme auf Quellenzeilen im Programm
- Single-Step-Ausführung und Trace-Back von C-Anweisungen
- Auswertung von Ausdrücken unter Verwendung von Programmvariablen

Eine vollständige und wirtschaftliche Lösung

Hiermit besteht eine vollständige und wirtschaftliche Lösung, den schon vorhandenen PC zusammen mit dem SSI Link & Locate-Paket und dem SoftProbe II/TX Debugger zur Entwicklung von eingebetteten Echtzeitanwendungen einzusetzen, die auf der 8086/186 Mikroprozessorfamilie basieren.

Eine vollständige Broschüre mit dem Titel »Writing ROMable Code in Microsoft C« ist erhältlich bei: Creative Daten Systeme GmbH, Bahnhofstr. 103, 8032 Gräfelfing, Tel.: (089) 854 30 80

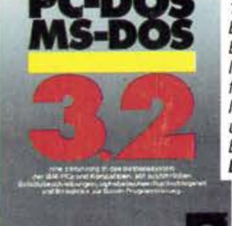
Kompaktes Wissen über Betriebs-systeme



H. H. Gerhardt: **DOS 3.3 für PCs und Personal System/2**
1988, 334 Seiten
Eine leichtverständliche Einführung mit Beispielen, Übungen und alphabetischer Befehlsübersicht. Für Anfänger und Fortgeschrittene. Mit umfangreichem Anhang: Begriffserklärung, Zeichencode-Tabellen, Kursdarstellung zum Thema »Netzwerk«, Fehlerübersicht, Übersicht zu den »DOS Function Calls« und »DOS Interrupt Calls«.
Bestell-Nr. 90547 ISBN 3-89090-547-1
DM 69,–/sFr 63,50/öS 538,20

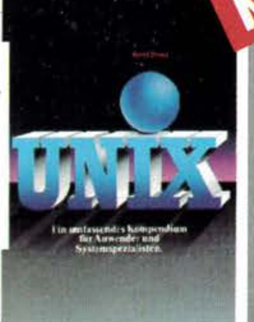
NEU

Peter Norton: **Peter Norton's Assemblerbuch**
1988, 352 Seiten, inkl. Diskette
Wenn Sie dieses Buch gelesen haben, wissen Sie, wie Sie vollständige Programme in Assemblersprache schreiben können: Texteditoren, Utilities und vieles mehr. Dabei lernen Sie eine Reihe von Techniken kennen, die von professionellen Programmierern verwendet werden!
Bestell-Nr. 90624, ISBN 3-89090-624-9
DM 79,–/sFr 72,20/öS 616,20



H. H. Gerhard: **PC-DOS/MS-DOS 3.2**
1987, 299 Seiten, inkl. Diskette
Eine Einführung in die wichtigsten Grundlagen zur Bedienung Ihres PCs mit DOS sowie ein hilfreiches Nachschlagewerk für jeden DOS-Anwender. Mit ausführlichen Befehlsbeschreibungen, alphabetischem Nachschlageteil und vielen Beispielen im PC-DOS- und MS-DOS-Format auf Diskette.
Bestell-Nr. 90519 ISBN 3-89090-519-6
DM 59,–/sFr 54,30/öS 460,20

NEU



H. Drees: **Unix**
1988, ca. 600 Seiten
Ein umfassendes Kompendium für Anwender und Systemspezialisten. Es beschreibt kein spezielles Unix, sondern Unix schlechthin und stützt sich dabei auf den Leistungsstandard, der durch Unix V repräsentiert wird. Mit vielen Beispielen und Anregungen.
Bestell-Nr. 90494 ISBN 3-89090-494-7
DM 89,–/sFr 81,90/öS 694,20



S. M. Sonner/M. Theis: **MS-OS/2 für Software-Entwickler**
1988, 474 Seiten
Dieses Buch bietet eine Einführung und einen Überblick über die theoretische Grundlagen von OS/2. Es wendet sich an Software-Entwickler und ein interessiertes Fachpublikum.
Bestell-Nr. 90638
ISBN 3-89090-638-9
DM 79,– /sFr 72,20/öS 616,20

Dr. Norbert Meder: **MS-OS/2**
1988, 304 Seiten
Einführung und Überblick über die neuen Programmiermöglichkeiten von MS-OS/2. Den Schwerpunkt bildet der neue Befehlsvorrat von MS-OS/2. Die einzelnen Befehle werden anhand von Beispielen leichtverständlich erläutert. Auch die Batch-Programmierung wird behandelt.
Bestell-Nr. 90512
ISBN 3-89090-512-9
DM 79,–/sFr 72,20/öS 616,20

U. Schmidt: **Das MS-Windows-Kompendium**
1988, ca. 350 Seiten, inkl. zwei Disketten
Tips, Tricks und Training, eine ausführliche Programmdokumentation und Hilfen bei der Anpassung verschiedener Softwarepakete finden Sie in diesem umfangreichen Kompendium. Für Windows 2.0 und 386. Auf den zwei Disketten finden Sie ein ausführliches Hilfsprogramm mit Installationsprogramm für die Festplatte, Utilities und zahlreiche Beispiele.
Bestell-Nr. 90558 ISBN 3-89090-558-7
DM 69,–/sFr 63,50/öS 538,20

NEU

Markt&Technik-Produkte erhalten Sie bei Ihrem Buchhändler, in Computer-Fachgeschäften oder in den Fachabteilungen der Warenhäuser.

Irrtümer und Änderungen vorbehalten.



Markt&Technik

Zeitschriften · Bücher
Software · Schulung

COMPUTER-LITERATUR UND SOFTWARE FRÜHJAHR/SOMMER '88

Fragen Sie Ihren Fachhändler nach unserem kostenlosen Gesamtverzeichnis mit über 500 aktuellen Computerbüchern und Software. Oder fordern Sie es direkt beim Verlag an!

Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-0.
SCHWEIZ: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Telefon (042) 415656.
ÖSTERREICH: Markt&Technik Verlag Gesellschaft m.b.H., Große Neugasse 28, A-1040 Wien, Telefon (0222) 5871393-0;
Rudolf Lechner&Sohn, Heizwerkstraße 10, A-1232 Wien, Telefon (0222) 677526;
Ueberreuter Media Verlagsges.mbH (Großhandel), Laudongasse 29, A-1082 Wien, Telefon (0222) 481543-0.

189/805